BACHELOR'S THESIS

Degree in Mathematics

Title: Algebraic phylogenetic reconstruction based on proteins and its application to *Persea americana*

Author: Raül Pérez i Gonzalo

Advisor: Marta Casanellas Rius and Jesús Fernández Sánchez

Department: Department of Mathematics

Academic year: 2017-2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Bachelor's Degree Thesis

Algebraic phylogenetic reconstrucion based on proteins and its application to *Persea americana*

Raül Pérez i Gonzalo

Supervised by: Marta Casanellas Rius
and Jesús Férnandez Sánchez

September, 2018

Abstract

Phylogenetic reconstruction methods attempt to infer the evolutionary relationships among a group of species. Apart from classical reconstruction methods, new methods have been developed based on algebraic relationships between the theoretical distribution of the molecular characters. All these methods have been only used so far for nucleotide sequences, as the SVD method. Moreover, the SVD method has been restricted to the reconstruction of quartet trees.

This thesis extends the SVD method to any number of character states (implemented in C++) and tests it with different scenarios of simulated amino-acid sequences. Next, SVD is incorporated into a supertree method, specifically the quartet-based method Weight Optimization (WO), to be able to reconstruct a tree with any number of species. In this case, SVD+WO is tested with sequences that have been simulated in analogy with a real data set. Moreover, the study of this data has allowed us to shed light on the controversial phylogeny of avocado (*Persea americana*).

Furthermore, all the simulated data are also reconstructed with a maximum likelihood software to compare the results obtained from the two reconstruction methods. Whereas the results obtained with our method SVD+WO are worse than those obtained by maximum likelihood, it is worth pointing out that our method can deal with much more general evolutionary models and takes less computation time.

Key words: phylogenetic invariants, topology reconstruction, heterogeneity across lineages, heterogeneity across sites, singular value decomposition, weight optimization

Acknowledgements

En primer lloc, voldria agrair tota la dedicació que han dedicat els meus tutors: el Jesús Fernández Sánchez i la Marta Casanellas. Agrair-los les hores dedicades (que han estat moltes), la seva orientació constant i les seves revisions, que han estat essencials. I gràcies per acollir-me tan bé.

A més, part d'aquest projecte s'ha fet amb la col·laboració del grup de recerca Evolutionary Genomics & Bioinformatics de la Universitat de Barcelona. Així, doncs, voldria donar les gràcies al Julio Rozas i a l'Alejandro Sánchez per haver-me guiat en tot moment i, sobretot, per ajudar-me quan sorgien els "grans" problemes.

Mencionar en especial al Xavi Ros, per tenir la capacitat de donar un cop de mà en tot el que calgui, i a l'Anna Pietx, per recolzament donat. Per últim, voldria agrair totes aquelles persones que m'han donat suport durant la realització del treball: a la Laurilla, al Jofre, al Cabanes, a la Ginesta, al Pau, a la Cristina, al Sergi, a la Paula, i, sense dubte, als meus pares.

Contents

Introduction **1**

1 Background **3**
1.1 Biological concepts 3
1.2 Singular Value Decomposition 6
1.3 Phylogenetic trees 10

2 Evolutionary models **13**
2.1 Markov models 14
2.2 Properties of evolutionary processes 16
2.3 Evolutionary processes on phylogenetic trees 18
2.4 Amino-acid models 20
2.4.1 PAM model 20
2.4.2 BLOSUM model 22
2.4.3 Other models 23

3 Reconstruction methods **25**
3.1 Maximum Likelihood 27
3.2 Singular Value Decomposition 29
3.3 SVD + Weight Optimization 34

4 Experimental analysis **40**
4.1 Study of SVD and ML on quartets 41
4.1.1 Simulated data on quartets 41
4.1.2 Reconstruction strategies on quartets 42
4.1.3 Reconstructing quartets: Results and discussion 43
4.2 Study of SVD+WO 47
4.2.1 Simulated data 47
4.2.2 Reconstruction strategies 55
4.2.3 Results and discussion 56

Conclusions **68**

REFERENCES **77**

Appendices **81**
A Auxiliary Figures 83
B Auxiliary Tables 91
C Implemented software 99

Introduction

Behind the huge variety of forms of life, the current organisms share several common features that can be used as important parameters to understand their origin. For example, from a chemical point of view, they are extremely similar, as nucleotides and amino-acids constitute the main biopolymers of all living organisms: the nucleic acids (RNA and DNA) and proteins. Basically, DNA carries the code to assemble proteins that are responsible for the main functions of an organism.

Furthermore, many evidences such as the universality of the genetic code and the universal conservation of multiple genes support that all current life on Earth have descended from a common ancestor [Koo03][GXL08] (as Charles Darwin predicted). As a result, the evolution of species can be mapped on a phylogenetic tree. The study of the evolutionary relationships among different species is called *phylogenetics*.

Phylogenetics is a relatively old area where morphological studies predate the availability of molecular sequences by several decades. The high new level of interest in this field in recent years is mainly a result of the availability of sequence data and of new methods for tree reconstruction.

Many tree inference methods have been proposed and the current state-of-the-art approach is to perform tree inference through a two-step process of multiple sequence alignment followed by statistical tree inference. The aim of the first step is to identify homologous characters between sequences and produce a heuristic estimate of those homologies in the so-called *alignment*. The second step, which is the problem we addressed in this project, typically uses only a single fixed alignment and a probabilistic evolutionary model to estimate the tree that best fits the observed sequences. This last step is achieved by the *phylogenetic reconstruction methods*.

Apart from the classical methods such as those based on genetic distances or on the full characters (maximum likelihood and Bayesian inference), other methods based on algebraic properties of the evolutionary models have been recently developed. More precisely, these methods profit from the properties of the so called *phylogenetic invariants*: polynomial relationships between the joint distribution of the observed characters that have evolved under an evolutionary model of molecular substitution. To date, these methods have been applied only to nucleotide sequences (four character states) and have shown their power in the reconstruction of quartet trees (trees with four species). However, these methods present certain advantages due to their flexibility, as they allow us to consider much more general models (for example, heterogeneous models across lineages).

The goal of this thesis is to develop a reconstruction method based on invariants for amino-acid sequences such that it can deal with a general number of species.
To this end, we have:

- Studied the mathematical background underlying the stochastic models of nucleotide and amino-acid substitution.
- Achieved a deep understanding of the algebraic framework that underlies the SVD method.
- Extended the code of Erik+2 ([CFS16]) to a general alphabet (any number of character states) and then we have applied some modifications due to the obtained results (see Remark 4.1.1). We called this adjustment *SVD* software.
- Customized the systems of weights of the Weight Optimization method (WO) to a new system of weights designed specifically for the SVD method: *SVD+WO*.
- Learned how to use several software of sequence simulation, and adapted them to our purposes.
- Designed different scenarios to test SVD and SVD+WO and compare the performance against a maximum likelihood method.
- Applied our tools to study a real data case (namely, we have shed light on the controversial phylogeny of avocado).

We have organized the report as follows. In the first chapter, we present some biological background, the basic low-rank approximation problem and an introduction to phylogenetic trees from the mathematical standpoint. After that, chapter 2 includes an overview of evolutionary models. In chapter 3, we describe the reconstruction methods that have been applied in our study. Finally, chapter 4 explains the implementation of the two programs and how the data have been generated. Moreover, this last chapter analyses the performance of our methods and of a maximum likelihood software, which allow us to benchmark their performance against an up-to-date phylogenetic reconstruction method.

It is worth noting that part of this project has been performed in conjunction with Evolutionary Genomics and Bioinformatics (EGB) group from the Universitat de Barcelona (UB).

1 Background

1.1 Biological concepts

Biological evolution is the set of changes in phenotypic and genetic characters of biological populations through generations. Natural selection was presented by Darwin as a possible explanation after observing the beak's variation of finches in the Galapagos Islands. Evolution's existence was confirmed showing the relationship between fossils and living species and, in addition, today evolution can be observed in real time in viruses.

Phylogenetics is the science that study the evolutionary relationships between different species or genes (different *taxa*). It represents the branching process of evolution in the so-called *phylogenetic tree*.

All living organisms store and replicate their information using nucleic acids, mostly with DNA (deoxyribonucleic acid) and some viruses with RNA (ribonucleic acid). Furthermore, this information is converted to proteins, which perform the vast biological functions, more precisely, the catalysts of chemical transformations. Then, proteins are the fundamental building blocks of life where natural selection acts.

These common properties of all organisms suggests that there is probably a last universal common ancestor (LUCA).

DNA structure. The DNA molecule is composed of two anti-parallel strands of nucleotides, which form a double helix. A nucleotide is a deoxyribose (five-carbon sugars) with a base linked with a phosphate group. The backbone of each strand is a repeating polymer chain of nucleotides, which are joined together by means of the link between the phosphate group and the sugar (see Figure 1.1).

The carbons in the sugar are numbered from 1 to 5 and the phosphate groups are linked to carbons 3 and 5. At one end or at the 5' end of each strand, the last carbon in the chain is a number 5 carbon, whereas at the other end there is the 3' end. By convention, every DNA sequence is written from 5' to 3', because this is the direction in which genetic information is transcribed.

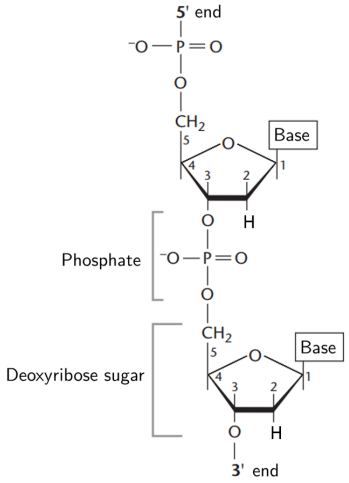


Figure 1.1: The join of two nucleotides is based on the link between a phosphate group to their deoxyribose units. Taken from [HA05].

In the DNA molecule, there are four types of bases, called adenine `A`, cytosine `C`, guanine `G` and thymine `T`. The two strands remain together by hydrogen bonds and are exactly complementary in sequence, so that where one has `A`, the other has `T` and, in the same way, with `C` and `G` (see Figure 1.2). Moreover, the two strands have opposite directions in reference to 5' and 3' end.

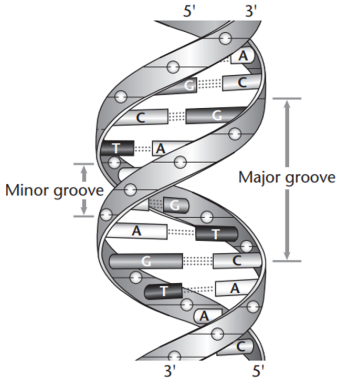


Figure 1.2: Schematic diagram of the DNA double helical structure. Taken from [HA05].

The backbone of RNA differs in that ribose sugars are used instead of deoxyribose. RNA is much more unstable than DNA, because the H on the second carbon in deoxyribose is now a OH group. Also, RNA is a single strand and the base uracil `U` occurs instead of `T`.

Protein structure. Proteins are linear polymers composed of chains of amino-acids. An amino-acid is composed by a central carbon or α-carbon that is linked to: a hydrogen, a carboxyl group (COOH), an amine group (NH_2) and a residue R.

There are 20 different amino-acids, which only differ in the residue group R. To form a protein, they are joined by a peptide bond, a link where the carboxyl group loses a hydrogen and oxygen and the amino group loses a hydrogen.

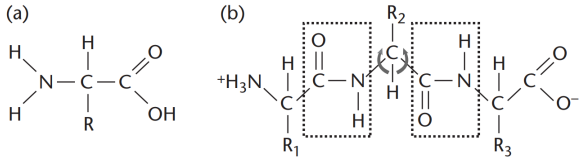


Figure 1.3: Chemical structure of an amino acid (a) and a protein backbone (b) composed of three amino-acids linked by peptide bonds (boxed). Taken from [HA05].

We will use the standard one-letter symbol of amino-acids to write a protein sequence (see Figure A.1 in the appendix). They are classified in four groups depending on their chemical properties and the combination between those groups establishes the protein folding (the formation of the three-dimensional structure of the protein) and its function.

Central Dogma of Molecular Biology. CDMB is the principle that claims that the information passes from DNA to RNA to proteins. More precisely, in the first step, from a functional unit of DNA sequence or the so-called *gene*, the information is converted in a messenger RNA (mRNA) and, then, in a second step, the ribosome reads triplets of nucleotides (called *codons)* to convert them into an amino-acid according to the *genetic code* (see Table B.1 in the appendix).

- In the first step, there is the synthesis of RNA, called *transcription*, where a DNA strand acts as a template. The process is based on forming a complementary chain, i.e., it will be the same as the non-template strand, except that where there would be a `T` (the template has an `A`), now there is a `U`.
- In the second step, there is the synthesis of a protein sequence, called *translation*, where now a mRNA acts as a template. The genetic information in the mRNA is coded in codons. Then, when part of the mRNA has been read, the next three nucleotides of the mRNA specifies the amino-acid that will be linked in the protein sequence.

The genetic code (Table B.1) shows which codons code for each amino acid. The fact that a group of codons contains the same specific amino acid information makes that the genetic code is redundant (degeneracy).

Sequence alignments. Throughout the lifetime of a cell, DNA information can be erroneously altered and then transmitted to daughter cells during cell division. These processes are known as *mutations* and are considered the main cause of evolution.

Due to CDMB, the replacement of a single nucleotide produces a change of the codon, which could code for another amino acid. Then mutations in the DNA molecule are the cause of protein mutations.

An *alignment* D is a way of arranging biological sequences (DNA, RNA, or protein) to identify regions of similarity that have evolved from a common ancestor. An alignment is represented within a matrix, where each row is an aligned sequence and each column represent the result of the evolutionary process of a single site (i.e., a nucleotide or an amino-acid).

Identifying regions that have evolved from a common ancestor is the first step in phylogenetics. These regions are called *homologous*. On the other hand, regions with high similarity, but without a common ancestor, are called *homoplasies*. Obviously only homologies are valid for establishing evolutionary relationships, then distinguish between homologous and homoplasies is essential in phylogenetics.

On the other hand, in the process of aligning the sequences, there are regions of a sequence that have not been identified as has evolved from a common ancestor of the others. For example, this may be due to an insertion of this region in the sequence or directly a mistake in the alignment. Then each column of the sequences that does not have these regions are represented by the - symbol and are called *gaps*.

Cajanus cajan	S K F P S - - - - - - - - - - - - E S
Vitis vinifera	S T A S T A N N V Q P C M K E R F N
Utricularia gibba	N G C G I L P A G E L R I G E L F M
Persea americana	F G L T V A A D A Q L C T E K L D T

Table 1.1: An example of a multiple sequence alignment D of four homologous proteins from species *Persea americana*, *Cajanus cajan*, *Utricularia gibba* and *Vitis vinifera*. This data was provided by the Avocado Genome Consortium.

1.2 Singular Value Decomposition

One of the strongest points of the SVD Theorem (see Theorem 1.2.6) is that it allows us to calculate the rank of a matrix with enough reliability. In fact, the last theorem of this section gives us much more than that: it gives us a measure of that reliability in terms of the distance from a matrix to the set of matrices of some given rank.

To this aim, some definitions and results must be introduced.

A good reference for the results of this section can be found in [Ste93] and [GL96]. From now on, we will adopt the convention that vectors u mean column vectors.

1.2.1 Definition (Matrix 2-norm) Given $M \in \mathbb{R}^{m\times n}$, the *2-norm* of M is defined by:

$$\|M\|_2 = \sup_{x\neq 0} \frac{\|Mx\|_2}{\|x\|_2}$$

where $\|x\|_2$ is the Euclidian-norm of the vector.

1.2.2 Remark It is clear that $\|M\|_2$ is the 2-norm of the largest vector obtained by applying M to a unitary 2-norm vector:

$$\|M\|_2 = \sup_{x\neq 0} \frac{\|Mx\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \|Mx\|_2 .$$

1.2.3 Definition (Matrix Frobenius-norm) With the same notation than in the previous definition, the *Frobenius-norm* or *F-norm* is defined by:

$$\|M\|_F = \sqrt{\sum_i \sum_j m_{ij}^2} = \sqrt{trace(MM^t)}$$

where m_{ij} are the entries of M.

1.2.4 Lemma The 2-norm and F-norm are invariant under orthogonal transformations of a matrix $M \in \mathbb{R}^{m\times n}$.

Proof. Let $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ be orthogonal matrices. The proof for the 2-norm is split into the following statements:

(i) $\|UM\|_2 = \|M\|_2$, for some $x \in \mathbb{R}^n$ such that $\|x\|_2 = 1$:
$$\|UM\|_2^2 = (UMx)^t(UMx) = x^tM^tU^tUMx = (Mx)^tMx = \|M\|_2^2.$$

(ii) $\|MV\|_2 = \|M\|_2$:
$$\|MV\|_2^2 = \sup_{\|x\|_2=1} \|MVx\|_2 = \sup_{\|Vx\|_2=1} \|MVx\|_2 = \sup_{\|y\|_2=1} \|My\|_2 = \|M\|_2^2.$$

In the case of the F-norm, the proof uses that given two matrices M_1 and M_2, we have that $trace(M_1M_2) = trace(M_2M_1)$:

$$\|UMV\|_F^2 = trace(UMV(UMV)^t) = trace(UMVV^tM^tU^t) = trace(UMM^tU^t) =$$
$$= trace(U^tUMM^t) = trace(MM^t) = \|M\|_F^2$$

□

For the following theorem, we need to introduce some notation. Given matrices M_1 and M_2, we denote by $[M_1, M_2]$ the matrix obtained by concatenating the columns of M_1 and M_2.

1.2.5 Lemma If $\bar{V}_1 \in \mathbb{R}^{n\times r}$ has orthonormal columns, then there exists $\bar{V}_2 \in \mathbb{R}^{n\times(n-r)}$ such that $V = [\bar{V}_1, \bar{V}_2]$ is orthogonal. Note that $< \bar{V}_1 >^{\perp} = < \bar{V}_2 >$, where $< \bar{V}_i >$ represents the space spanned by the column vectors in $\bar{V}_i$.

Proof. This is a consequence of the Gram-Schmidt algorithm.

□

1.2.6 Theorem (Singular Value Decomposition) Let M be a $m \times n$ matrix with real entries, then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that

$$U^tMV = D \ , \quad \text{with } D \in \mathbb{R}^{m\times n} \text{ and } D = diag\,(\sigma_i) \ , 1 \le i \le l = min(m,n)$$

where $\sigma_1 \ge \sigma_2 \ge ... \ge \sigma_l \ge 0$.

Proof. We denote $\sigma = \|M\|_2$. Let $x \in \mathbb{R}^n$ be a unitary vector in 2-norm such that $\|Mx\|_2 = \|M\|_2 = \sigma$ and let $y \in \mathbb{R}^m$ be the unitary vector such that $Mx = \sigma y$ (*).

From the previous lemma, there exist $\bar{V} \in \mathbb{R}^{n\times(n-1)}$ and $\bar{U} \in \mathbb{R}^{m\times(m-1)}$ such that $V = [x, \bar{V}] \in \mathbb{R}^{n\times n}$ and $U = [y, \bar{U}] \in \mathbb{R}^{m\times m}$ are orthogonal. Write $\bar{u}_1, \ldots, \bar{u}_{n-1}$ for the columns of the matrix $\bar{U}$.

Using (*) and the orthogonality properties, $y^tMx = \sigma y^t y = \sigma$ and $\bar{u}_i^t Mx = \sigma \bar{u}_i^t y = 0\ \forall i \in \{1, ..., n-1\}$, so we have the following equality:

$$\tilde{M} := U^t M V = \begin{pmatrix} \sigma & w^t \\ 0 & B \end{pmatrix} \Rightarrow \tilde{M} \begin{pmatrix} \sigma \\ w \end{pmatrix} = \begin{pmatrix} \sigma^2 + w^t w \\ Bw \end{pmatrix}.$$

From this,

$$\left\| \tilde{M} \begin{pmatrix} \sigma \\ w \end{pmatrix} \right\|_2^2 = (\sigma^2 + w^t w)^2 + \|Bw\|_2^2 \geq (\sigma^2 + w^t w)^2.$$

By the definition of 2-norm matrix, we have $\left\|\tilde{M}\right\|_2^2 \geq (\sigma^2 + w^t w)$. On the other hand, as the 2-norm is invariant under orthogonal transformations, we also have that $\|M\|_2 = \left\|\tilde{M}\right\|_2$. All together gives that $\sigma^2 = \|M\|_2^2 = \left\|\tilde{M}\right\|_2^2 \geq (\sigma^2 + w^t w) \Rightarrow w = 0$. Iteratively doing the same, now with the B matrix, we get the $n-1$ remaining σ_i. Hence, $\tilde{M} = D = diag(\sigma_i)$. □

1.2.7 Definition (Singular values) Keeping the notation as in the previous theorem, if $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_l = 0$, then $\sigma_1, \sigma_2, ..., \sigma_r$ are called the *singular values* of the matrix M.

1.2.8 Corollary The number of singular values equals the rank of the matrix.

Proof. It is an immediate consequence that the rank of a matrix does not change if we multiply it by invertible matrices. □

1.2.9 Corollary If M has the singular values $\sigma_1 \geq ... \geq \sigma_r$ then

$$\|M\|_2 = \sigma_1 \quad \text{and} \quad \|M\|_F = \sqrt{\sigma_1^2 + ... + \sigma_r^2}.$$

Proof. As $M = UDV^t$, by virtue of Lemma 1.1.4, we have that

$$\begin{cases} \|M\|_F = \|D\|_F = \sqrt{\sigma_1^2 + ... + \sigma_r^2} \text{ by definition,} \\ \|M\|_2 = \|D\|_2 . \end{cases}$$

To prove that $\|D\|_2 = \sigma_1$, without loss of generality, we take $m \geq n$. For any given unitary 2-norm vector $x \in \mathbb{R}^n$, we have

$$\|Dx\|_2 = \sqrt{\sigma_1^2 x_1^2 + ... + \sigma_n^2 x_n^2} \leq \sigma_1 \sqrt{x_1^2 + ... + x_n^2} = \sigma_1 \|x\|_2 = \sigma_1.$$

Moreover, if $x = (1, 0, ..., 0)$, this inequality becomes an equality. It follows that

$$\sigma_1 = \max_{\|x\|_2 = 1} \|Dx\|_2 = \|D\|_2 .$$

□

Thereby, given a set of matrices $\{M_1, M_2, ..., M_N\}$ it is usual to be interested in knowing which one is closest to have a given rank. To face this problem we will introduce two distances.

Firstly, it is necessary to see that the decomposition of M into singular values gives us an especially useful way of writing M as the sum of matrices of rank 1.

1.2.10 Proposition Keeping the notation as above, we have the following equality of matrices

$$M = \sum_{i=1}^{r} \sigma_i u_i v_i^t$$

where $U = [u_1 \ ... \ u_m]$, $V = [v_1 \ ... \ v_n]$ and $r = rank(M)$.

Proof. Let us split $D = D_1 + ... + D_r$, where each $D_i \in \mathbb{R}^{m\times n}$ is the matrix with all entries equal to zero, except for that in the i-th row and i-th column, which is σ_i, i.e., $D_i = diag(0, ..., 0, \sigma_i, 0, ..., 0)$. It is immediate that

$$M = \sum_{i=1}^{r} UD_iV^t \qquad \text{with } \ UD_iV^t = \sigma_i u_i v_i^t.$$

□

Theorem 1.2.11 and 1.2.12 are known as **Eckart-Young-Mirsky Theorem**, but here it is explained separately:

1.2.11 Theorem (2-distance of M to lower-rank matrices) Let $M = UDV^t$ be a SVD of the matrix M and define $E_k = \{\bar{M} \in \mathbb{R}^{m\times n} \mid rank(\bar{M}) \leq k\}$. Let M_k be the projection of M into E_k: $M_k = \sum_{i=1}^{k} \sigma_i u_i v_i^t$. Then:

$$d_2(M, E_k) = \min_{rank(\bar{M})\leq k} \left\|M - \bar{M}\right\|_2 = \|M - M_k\|_2 = \sigma_{k+1}$$

Proof. From the SVD of the matrix M, it is immediate to obtain the SVD of the matrices M_k and $M - M_k$:

$$\begin{cases} U^t M_k V = diag(\sigma_1, ..., \sigma_k, 0, ..., 0) \\ U^t (M - M_k) V = diag(0, ..., 0, \sigma_{k+1}, ..., \sigma_l). \end{cases}$$

By corollary 1.2.9, it follows that $\|M - M_k\|_2 = \sigma_{k+1}$. To prove that σ_{k+1} is the value of the 2-distance, it is enough to check that $\|M - M_k\|_2 \leq \left\|M - \bar{M}\right\|_2 \ \forall \bar{M} \in E_k$.

For any $\bar{M} \in E_k$, as $rank(\bar{M}) \leq k$, we can find orthonormal vectors $\{x_1, ..., x_{n-k}\} \in \mathbb{R}^n$ so that $< x_1, ..., x_{n-k} > \subset$ Ker $(\bar{M})$.

By Grassmann formula, it is easy to derive that $< x_1, ..., x_{n-k} > \cap < v_1, ..., v_{k+1} > \neq \{0\}$, so there exist a unitary 2-norm vector $z \in \mathbb{R}^n$ in this intersection. In particular $\bar{M}z = 0$, by Proposition 1.2.10

$$Mz = \sum_{i=1}^{k+1} \sigma_i (v_i^t z) u_i.$$

Thereby, we finally have

$$\left\|M - \bar{M}\right\|_2^2 \geq \left\|(M - \bar{M})z\right\|_2^2 = \|Mz\|_2^2 = \left\|\sum_{i=1}^{k+1} \sigma_i (v_i^t z) u_i\right\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^t z)^2 \geq \sigma_{k+1}^2$$

where in the last equality we have used that $\{u_1, ..., u_m\}$ are an orthonormal basis of $\mathbb{R}^m$ and $\|Vz\|_2^2 = \sum_{i=1}^{k+1} (v_i^t z)^2 = 1$, as V is an orthogonal matrix.

□

Now, the analogous theorem for Frobenius-norm is presented:

1.2.12 Theorem (F-distance of M to lower-rank matrices) Keeping the notation as in the above theorem, Theorem 1.2.11, then

$$d_F(M, E_k) = \min_{rank(\bar{M}) \leq k} \left\|M - \bar{M}\right\|_F = \|M - M_k\|_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_r^2}.$$

Proof. We proceed as in the proof of the previous result. Note that we have $\|M - M_k\|_F = \sqrt{\sigma_1^2 + ... + \sigma_r^2}$.

Therefore, we only need to show $\|M - M_k\|_F \leq \left\|M - \bar{M}\right\|_F \ \forall \bar{M} \in E_k$.

From now on, we denote $\sigma_i(M)$ the i-th singular value of M if $i \in 1, ..., r$, else if $r < i \leq l$, we have $\sigma_i(M) = 0$.

Given $M', M'' \in \mathbb{R}^{m \times n}$ such that $M = M' + M''$, the triangle inequality states that $\sigma_1(M) = \|M\|_2 \leq \|M'\|_2 + \|M''\|_2 = \sigma_1(M') + \sigma_1(M'')$.

Now, respectively for M' and M'', suppose M_i' and M_i'' denote the projection to E_i. Then, $\sigma_i(M') = \sigma_1(M' - M_{i-1}')$ and $\sigma_i(M'') = \sigma_1(M'' - M_{i-1}'')$.

For any $i, j \geq 1$ such that $i + j - 1 \leq l = min(m, n)$, we have that

$$\begin{aligned} \sigma_i(M') + \sigma_j(M'') = \sigma_1(M' - M_{i-1}') + \sigma_1(M'' - M_{j-1}'') \geq \\ \geq \sigma_1(M - M_{i-1}' - M_{j-1}'') \geq \sigma_{i+j-1}(M). \end{aligned}$$

Notice that if $\bar{M} \in E_k$, then $\sigma_{k+1}(\bar{M}) = 0$. So, taking $M' = M - \bar{M}$ and $M'' = \bar{M}$, as $M = M' + M''$, we conclude that for $i \geq 1$, $j = k + 1$

$$\sigma_i(M - \bar{M}) + \sigma_{k+1}(\bar{M}) = \sigma_i(M - \bar{M}) \geq \sigma_{k+i}(M).$$

Therefore, the claims follows:

$$\left\|M - \bar{M}\right\|_F^2 = \sum_{i=1}^{r} \left(\sigma_i(M - \bar{M})\right)^2 \geq \sum_{i=k+1}^{n} \left(\sigma_i(M)\right)^2 = \|M - M_k\|_F^2.$$

□

1.3 Phylogenetic trees

A tree $\mathcal{T}$ is a connected acyclic graph. Among the vertices of the tree, we distinguish between the terminal nodes, the *leaves*, and the interior nodes. The *degree* of a node in a graph is the number of edges adjacent to it. The leaves of a tree are the nodes whose degree are one. We denote by $\mathcal{L}(\mathcal{T})$ the set of leaves, by $E(\mathcal{T})$ the set of edges and by $N(\mathcal{T})$ the set of all nodes of $\mathcal{T}$.

According to [AR04] and [AR05], a phylogenetic tree records the evolutionary history of a set of current species. More precisely,

1.3.1 Definition (Phylogenetic tree) A *phylogenetic tree* is a triplet $(\mathcal{T}, X, \phi)$ where $\mathcal{T}$ is a tree with n leaves, X is a set of different (usually current) biological species, and $\phi : X \longrightarrow \mathcal{L}(\mathcal{T})$ is a bijection. That is, in a phylogenetic tree, the leaves are labeled by the set X. While the internal nodes represent possibly extinguished species, the leaves represent the current species.

1.3.2 Definition (Rooted phylogenetic tree) A *rooted phylogenetic tree* is a tree $\mathcal{T}$ where a distinguished interior vertex is labelled to be the *root* r. The root represents the last common ancestor of the species in X. It induces an orientation of the edges of $\mathcal{T}$ in the direction in which time has elapsed.

Given a pair of nodes connected directly with an edge, the *parent* node is that which lies closer to the root, while the other is the *child*.

A rooted tree is *binary* if its root has degree two and all other interior nodes have degree three. For an unrooted tree, we say that is *trivalent* if all its interior nodes have degree three. Basically, we will work with these two types of trees.

1.3.3 Example An example of a phylogenetic tree in a largest scale is the tree of life:

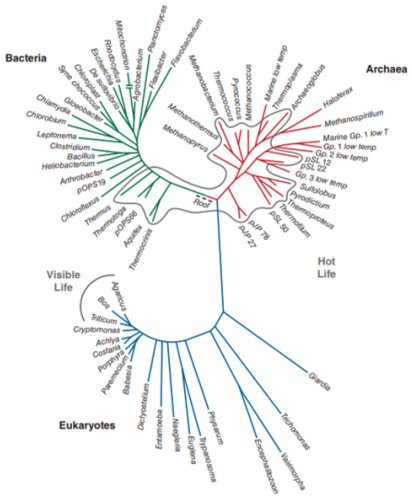


Figure 1.4: The tree of life. The three major branches represent the three domains proposed by Woese (1990): bacteria, archaea and eukaryotes. Taken from [Pev15].

In a phylogenetic tree two kinds of information are represented. First, we have the tree topology, which describes the evolutionary relationships between the species and corresponds to the structure of the labeled graph (see Definition 1.3.4). Next, we have the branch lengths (i.e., the length of the edges), which represent the evolutionary time in terms of the amount of elapsed substitutions per site between the ends of the edge.

1.3.4 Definition (Tree topology) Let $(\mathcal{T}_1, X, \phi_1)$ and $(\mathcal{T}_2, X, \phi_2)$ be two phylogenetic trees with the same set of leaves. We say that they have the same *tree topology* if there exists a homeomorphism $f : \mathcal{T}_1 \longrightarrow \mathcal{T}_2$ such that

$$f(\phi_1(x_i)) = \phi_2(x_i) \ \ \forall i = 1, ..., n.$$

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are rooted and r_1, r_2 are their respective roots, we will also impose that $f(r_1) = r_2$.

1.3.5 Examples In Figure 1.5 it is shown the different tree topologies of an unrooted tree of 4 leaves. They are denoted from left to right by 12 | 34, 13 | 24 and 14 | 23.

Figure 1.5: The three possible topologies of an unrooted tree $\mathcal{T}$ of four leaves.

Unrooted trees are less informative than rooted ones. Then, it is necessary a prior knowledge to convert unrooted trees to rooted trees. For example, a usual strategy is to introduce in the analysis the earliest branching species in a tree, which is called an *outgroup*. Then we can root the tree on the branch connecting the outgroup to the rest of the species.

Finally, in order to describe the computational complexity of finding the correct tree in a phylogenetic reconstruction process, we will need some combinatorics (see [PS05]). For a trivalent tree with n leaves, there are $2n-3$ edges. The number of distinct tree topologies of trivalent unrooted trees on n leaves is $(2n-5)!!$, while the number of rooted tree topologies is $(2n-3)!!$.

2
Evolutionary models

Changes in the DNA molecule are the main cause of the evolution of species, but these changes can be due to mutations, deletions, insertions, translocations or even to horizontal transfers, when the new genetic material does not come from its ancestor. In order to design and use phylogenetic reconstruction methods and because of evolution complexity, it is necessary to model evolutionary processes (a look to [Gas05] is highly recommended for further details on modeling evolution).
To this aim, it is usual to adopt the following assumptions:

(1) We will only consider DNA mutations as the cause of the evolutionary process. Moreover, these mutations occur randomly.

(2) Evolutionary processes at different adjacent edges only depend on the common node, that is, they follow a Markov process (see Definition 2.1.3 below).

(3) Each site in the DNA chain mutates independently of the others and under the same probabilities of changing. Thus, given a nucleotide sequence $s_1^p s_2^p ... s_n^p$, the probability of mutating another sequence $\mathcal{S}_c = s_1^c s_2^c ... s_n^c$ is

$$P(\mathcal{S}_p \to \mathcal{S}_c) = \prod_{i=1}^{n} P(s_i^p \to s_i^c).$$

Thanks to CDMB, accepting these assumptions for nucleotide sequences implies that they are also fulfilled in the amino-acid level.

These assumptions are not realistic and may be possible reasons of wrong phylogenetic inference. On the other hand, although assumption (1) can be seen as a major restriction, the results obtained in phylogenetic inference are fairly good ([OR07]) if we start with a good alignment.

Finally, the assumption (3) is equivalent to say that all the positions in the sequence are independent and identically distributed, or that they follow the *i.i.d. hypothesis*, so it allows us modeling the whole evolutionary process of a sequence by studying a single position. For the purpose of relaxing this hypothesis, *mixture models* consider different stochastic models in the same sequence (see forthcoming Section 2.3).

2.1 Markov models

2.1.1 Definition (Markov matrix) Given a square matrix M of size k, it is called a *Markov matrix* (or stochastic matrix) if their entries $m_{ij} \in [0,1]$ and for all $i = 1, ..., k$ $\sum_{j=1}^{k} m_{ij} = 1$.

2.1.2 Remark In phylogenetic studies, the columns and rows of Markov matrices are usually labeled by the nucleotides ($k = 4$) or by the amino-acids ($k = 20$). The usual convention in literature for transition matrices of amino-acids is that columns sum to 1. To be consistent with our presentation of Markov matrices, we will work with the transpose of these matrices so that rows sum to 1 (as in the nucleotide case).

2.1.3 Definition (Discrete-time probabilistic process) Taking time as a discrete variable and representing it by time steps t_i with $i \in \{0, 1, ..., n\}$, we consider a *discrete-time probabilistic process* as a family of random variables $(X_{t_0}, X_{t_1}, ..., X_{t_n})$, where each X_{t_i} takes values in the same space of states (nucleotides or amino-acids).

2.1.4 Definition (Markov process) A discrete-time probabilistic process is a *Markov process* if the future is independent of the past given the present, i.e.:

$$P(X_{t_{n+1}}|X_{t_0}, X_{t_1}, ..., X_{t_n}) = P(X_{t_{n+1}}|X_{t_n}).$$

As, in general, time is unknown in phylogenetic inference, from now on we will write X_i, where this random variable represents the nucleotide or amino-acid at the i-th step. Thus, X_i takes values/states in the set $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}$ or, in the case of amino-acids, $\Sigma = \{\mathtt{A}, \mathtt{R}, \mathtt{N}, ..., \mathtt{W}, \mathtt{Y}, \mathtt{V}\}$.

Given a general alphabet of $\Sigma = \{\varsigma_1, ..., \varsigma_\mathtt{k}\}$ with k states, to establish an evolutionary process it is necessary to determine the parameters of the Markov model. Specifically, we represent by an edge e the evolutionary process between the original sequence and a mutated version of it: $X_r \xrightarrow{e} X_c$. Then, we can define:

2.1.5 Definition (Substitution or Transition Matrix) A *transition matrix* (or *substitution matrix*) associated to an evolutionary process e is a $k \times k$ Markov matrix S_e with the following form:

$$S_e(t) = \begin{array}{c} \\ \varsigma_1 \\ \varsigma_2 \\ \vdots \\ \varsigma_\mathtt{k} \end{array} \begin{array}{c} \begin{array}{cccc} \varsigma_1 & \varsigma_2 & \cdots & \varsigma_k \end{array} \\ \begin{pmatrix} P(\varsigma_1|\varsigma_1,t) & P(\varsigma_2|\varsigma_1,t) & \cdots & P(\varsigma_k|\varsigma_1,t) \\ P(\varsigma_1|\varsigma_2,t) & P(\varsigma_2|\varsigma_2,t) & & \\ \vdots & & \ddots & \vdots \\ P(\varsigma_1|\varsigma_k,t) & & \cdots & P(\varsigma_k|\varsigma_k,t) \end{pmatrix} \end{array}.$$

Each entry $P(\varsigma_j|\varsigma_i, t)$ is the conditional probability that the state ς_i at the original sequence of e is being substituted by the state ς_j after some time t, i.e., $P(\varsigma_j|\varsigma_i, t) = P(X_c = \varsigma_j|X_p = \varsigma_i, t)$.

2.1.6 Remark Note that, effectively, S_e is a Markov matrix, because:

$$\sum_{j=1}^{k} m_{ij} = \sum_{j=1}^{k} P(\varsigma_j|\varsigma_i, t) = 1.$$

2.1.7 Definition (State distribution at X_r) As above, given the evolutionary process $X_r \xrightarrow{S_e} X_c$, we denote by $\pi^r = (\pi^r_{\varsigma_1}, \ldots, \pi^r_{\varsigma_k})$ the *state distribution at the ancestral sequence* X_r, where each $\pi^r_{\varsigma_i}$ is the probability of observing ς_i in X_r. In particular, all entries of π^r are nonnegative and $\sum_{i=1}^{k} \pi^r_{\varsigma_i} = 1$.

If no further restrictions are imposed to S_e and π^r, we have the general Markov model (GMM). In inferring phylogenies, we can work with less complex models (few parameters), but this may influence the reconstruction performance, since more complex models come closer to reality but make them less tractable.

So far, we have been working with discrete-time probabilistic processes, but if we adopt a continuous-time perspective, we can give a new approach to evolutionary processes:

2.1.8 Definition (Rate matrix) For each pair of $\varsigma_j, \varsigma_j \in \Sigma$, the instantaneous rate of substitution at time t_0 is:

$$q_{\varsigma_i\varsigma_j}(t_0) = \lim_{t \to t_0} \frac{P(\varsigma_j|\varsigma_i, t) - P(\varsigma_j|\varsigma_i, t_0)}{t - t_0}.$$

These values are to be taken as the entries of a *rate matrix* $Q_e(t_0) = Q(t_0)$. More generally, a square matrix Q is called a *rate matrix* if $q_{\varsigma_i\varsigma_j}(t_0) = -\sum_{\varsigma_j \neq \varsigma_i}^{k} q_{\varsigma_i\varsigma_j}(t_0)$, where $q_{\varsigma_i\varsigma_j}(t_0) \geq 0 \ \forall \varsigma_i \neq \varsigma_j$.

2.1.9 Proposition Assuming that the values $q_{\varsigma_i\varsigma_j}(t_0)$ do not depend on t_0 (time-homogeneous assumption along the edge e), the resulting rate matrix Q has the following properties:

1) Q matrix is singular ($\det(Q) = 0$).
2) Q is a rate matrix $\iff$ e^{Qt} is a Markov matrix $\forall t \geq 0$ ([PS05]).
3) $S(t) = e^{Qt}$ is the unique solution to $S'(t) = S(t)Q$, with $S(0) = Id$.
4) Chapman-Kolmogorov equation is fulfilled: $S(t + s) = S(t)S(s)$.

2.1.10 Remark As e^{Qt} is invertible, not all Markov matrices can be written as $S(t) = e^{Qt}$. Here we have a counterexample:

$$S(t) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

2.1.11 Remark [KG05] When a Markov matrix S can be written as $S = e^Q$ and it is diagonalizable, eigen decomposition is the standard way to compute Q.

Indeed, if S is diagonalizable, write $S = UDU^{-1}$ where $D = diag(d_i)$ is diagonal. Consider $Q = U\ln(D)U^{-1}$ where $\ln(D) = diag(\ln d_i)$. Using the power series development of the exponential, we know that $e^{UVU^{-1}} = Ue^VU^{-1}$. Therefore

$$e^Q = e^{U\ln(D)U^{-1}} = Ue^{\ln(D)}U^{-1} = S$$

Note that if $S = e^{tQ}$ is not diagonalizable, it can still be written as $S = UJU^{-}1$, where J is a Jordan block matrix. In any case, S can be approximated by using the Taylor expansion of the exponential function applied to tQ:

$$S(t) = Id + tQ + \frac{(tQ)^2}{2!} + ... + \frac{(tQ)^n}{n!} + ...$$

From now on, time is measured by the expected number of changes per site (*mutation rate* μ) rather than using astronomical time units. With this convention, we can write any rate matrix as μQ, where $-\sum_{i=1}^{k} q_{\varsigma_i\varsigma_i}\pi^r_{\varsigma_i} = 1$. Such a Q is called a *normalized rate matrix*.

For normalized rate matrices, the average rate of replacement at equilibrium is one. Therefore, the mutation rate μ appears as a new parameter, which provides the possibility to increase or decrease the mean rate: $S_e(t) = e^{\mu_e Q_e}$.

In the following sections several definitions and facts of evolutionary models and processes are presented, which are decisive for phylogenetic reconstruction.

2.2 Properties of evolutionary processes

2.2.1 Definition (Stationarity) A Markov evolutionary process $X_r \to X_c$ is *stationary* when as time t goes to infinity, the probability of observing any state ς_i is non-zero and independent of the state distribution at π^r, i.e., independent of the starting point. In this case, for each state ς_i, we denote

$$\pi_{\varsigma_i} = \lim_{t\to\infty} P(\varsigma_i|t) \in (0, 1].$$

The vector $\pi = (\pi_{\varsigma_1}, ..., \pi_{\varsigma_k})$ is called the *stationary distribution* of the process. Moreover, it is satisfied for any $\varsigma_i \in \Sigma$ that

$$\pi_{\varsigma_i} = \lim_{t\to\infty} \sum_{j=1}^{k} \pi_{\varsigma_j} P(\varsigma_i|\varsigma_j, t).$$

2.2.2 Definition (Time-reversibility) A Markov process is *time-reversible* if the probability of sampling ς_i from the stationary distribution π and changing to state ς_j is the same as the probability of sampling ς_j from the stationary distribution and changing to state ς_i. That is, for all $\varsigma_i, \varsigma_j \in \Sigma$ and $t \geq 0$ we have

$$\pi_{\varsigma_i} P(\varsigma_j|\varsigma_i, t) = \pi_{\varsigma_j} P(\varsigma_i|\varsigma_j, t).$$

2.2.3 Remark Under the homogeneous continuous assumption, time-reversibility implies that the rate matrix Q can be written as

$$Q = RD(\pi)$$

where R is a symmetric matrix and $D(\pi)$ is the diagonal matrix that has the values of the stationary distribution in its diagonal. For example, for DNA alphabet, the general Time-Reversible (GTR, [Tav86]) model has the following structure

$$Q = \begin{pmatrix} \cdot & \alpha\pi_C & \beta\pi_G & \gamma\pi_T \\ \alpha\pi_A & \cdot & \delta\pi_G & \epsilon\pi_T \\ \beta\pi_A & \delta\pi_C & \cdot & \zeta\pi_T \\ \gamma\pi_A & \epsilon\pi_C & \zeta\pi_G & \cdot \end{pmatrix},$$

where dots represent the convenient values so that rows of Q sum to 0.

As claimed in the beginning of this chapter, it is quite unrealistic that different sites in a sequence evolve equally under the same rates and independently (*i.i.d. hypothesis*). To close this section, we deal with the possibility of relaxing this assumption. As a result, we can cope with real data sequences, which may have fast and slow evolving sites, mostly as a consequence of variable selective pressure. For example, it is known that functionally important sites are conserved during evolution, while unimportant sites are free to vary.

2.2.4 Definition (Homogeneity across sites) Given a sequence evolving under a Markov process, *homogeneity across sites* is the assumption that all sites in the sequence evolve under the same probabilities/rates. Otherwise, we have a *heterogeneous model across sites*.

In order to face the possibility of a heterogeneous model across sites, there are different strategies.

Firstly, to deal with different evolving rates, they are approximated by a *continuous probability distribution* ([LSV09]). Then, given a sequence $\mathcal{S}$ and a model $\mathcal{M}$, the probability for a site s_i being $\varsigma \in \Sigma$ is calculated by integrating over all possible rates:

$$P(\mathrm{s}_i = \varsigma) = \int_0^\infty P(\mathrm{s}_i = \varsigma \mid \mu(i) = \mu, \mathcal{M}) f(\mu)\, d\mu$$

where f is the probability density of the assumed rate distribution, and where $P(\mathrm{s}_i = \varsigma \mid \mu(i) = \mu, \mathcal{M})$ is the probability for character s_i conditional on rate $\mu(i) = \mu$ for this site. Note that instead of assigning sites to rate classes, each site considers all possible assignments in the probability calculation.

The density function f is usually approximated by a Γ distribution. Although there is not a particular reason for the distribution of rates in a real sequence to follow a Γ distribution, this function with only one parameter is sufficiently flexible to fit real data quite well ([Yan96]). It is said, then, that it follows a model $\mathcal{M} + \Gamma$.

Since the numerical calculation of this last integral is computationally expensive, it is used a *discrete distribution* of rates across sites, usually a discretized Γ distribution:

$$P(\mathrm{s}_i = \varsigma) = \sum_{j=1}^{g} P(\mathrm{s}_i = \varsigma \mid \mu(i) = \mu_j, \mathcal{M}) p_j$$

where g is the number of rate classes and p_j the probability of the rate class j.

2.3 Evolutionary processes on phylogenetic trees

From now on, we assume that $\mathcal{T}$ is a (rooted) binary tree, where its edges represent evolutionary processes. We will denote by $\mathcal{M}_e = \{S_e, \pi^r\}$ the model parameters of the edge e if e emerges from the root r, or $\mathcal{M}_e = \{S_e\}$ for the other edges. Then $\mathcal{M} = \{\{S_e\}_{e \in E(\mathcal{T})}, \pi^r\}$ represents the evolutionary model of $\mathcal{T}$.

The complexity of the heterogeneity across sites hypothesis can be increased if it is assumed that distinct regions of a sequence have evolved independently, but following different evolutionary processes, or even, different phylogenetic trees. This possibility is taken into account in the *mixture models*. For example, if our sequence is the concatenation of two genes or proteins such that one appears by horizontal transfer, they will probably follow different phylogenetic trees and models.

2.3.1 Definition (r-mixture) An alignment follows an *r-mixture* if it can be split into r partitions, where each partition i has evolved according to a different evolutionary model $\mathcal{M}^i$ on a phylogenetic tree $\mathcal{T}_i$.

Then, given an alignment D, we can split $D = D_1 \cup D_2 \cup ... \cup D_r$, where each D_i has evolved under $(\mathcal{T}_i, \mathcal{M}^i)$.

As a particular case, mixture models allow us the possibility of having *invariant sites*, that is, sites where we always observe the same state independently of time. This possibility is denoted by +I. The corresponding evolutionary process is described by a null rate matrix, i.e., the transition matrix is Id.

Thanks to sequencing technology, we can access to genome information, allowing us to know the values of the random variables at the leaves, but predecessor species do not exist anymore, making random variables of internal nodes unknown.

2.3.2 Example In Figure 2.1, X_1, X_2, X_3 and X_4 represent the observable variables and X_5, X_6 and X_7 the hidden ones. According to our assumptions, $\mathcal{T}$ follows a Markov process, so we have that X_1 and X_2 only depend on X_5, which only depends on X_7.

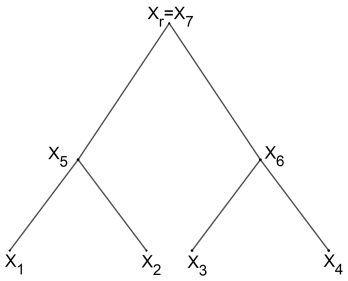


Figure 2.1: A rooted tree $\mathcal{T}$ of four leaves $X_1, ..., X_4$.

All these conditions lead to the so-called *Hidden Markov Model* (HMM), which we adopt from here on.

2.3.3 Definition (Hidden Markov Model) Given a phylogenetic tree that evolves under Markov process, each evolutionary process $X_i \xrightarrow{S_e} X_j$ is ruled by a Markov matrix. If the states of some random variables of the Markov process are unknown, it is called a *Hidden Markov Model* (HMM).

2.3.4 Definition (Homogeneity across lineages) *Homogeneity across lineages* on a tree $\mathcal{T}$ is the assumption that all the evolutionary processes associated with the edges of the tree are ruled by the same rate matrices, that is, $Q_e = Q$ for all edge e in $\mathcal{T}$. Otherwise, we have a *heterogeneous model across lineages.*

2.3.5 Example In Figure 2.2 is represented a phylogenetic tree with a heterogeneous model across lineages, as the evolutionary process of each edge in the tree follows a different model.

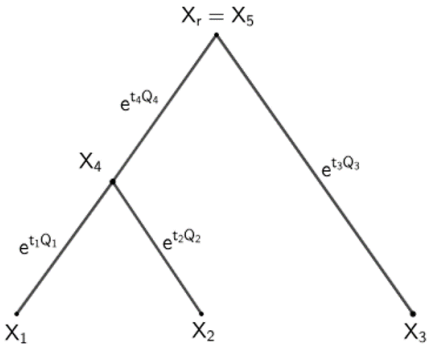


Figure 2.2: A heterogeneity across lineages model on a rooted tree $\mathcal{T}$ of three leaves. Transition matrices for each evolutionary process are indicated along the branches.

2.4 Amino-acid models

Evolutionary models based on amino-acid suppose a large increase in the number of parameters compared to those based on nucleotides. Thereby, while in nucleotide models we can infer parameters such as the mutation rates or mutation probabilities from the data, amino-acids models are constrained to certain given rate matrices, which have been previously designed empirically from large databases. Furthermore, all these models were performed under the stationarity assumption with the stationary distribution π being equal to the state distribution π^r.

These models attempt to summarize the biological, chemical and physical relationships between amino acids. Firstly, some basic concepts are introduced:

2.4.1 Definition (Accepted point mutation) An *accepted point mutation* or *PAM* is a replacement of a single amino-acid in a protein by another residue. This new residue is usually taken as one that has been already observed in real data, as accepted by natural selection.

2.4.2 Definition (Multiple substitution) Given a pair of sequences, we say that there is a *multiple substitution* in a site if we observe less evolutionary changes (accepted point mutations) that those that have really occurred.

2.4.3 Examples Assume we have two evolutionary processes from the same original sequence $\mathcal{S}_0$, giving to a pair of sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ at the leaves of a tree.

1) If a site in sequence $\mathcal{S}_0$ evolves from $\mathtt{A} \to \mathtt{D} \to \mathtt{B}$ in sequence $\mathcal{S}_1$, we observe one accepted point mutation, while the real number of mutations is two. In this case, we have a multiple substitution.

2) If a site in sequence $\mathcal{S}_0$ evolves from $\mathtt{A} \to \mathtt{D} \to \mathtt{A}$ in sequence $\mathcal{S}_1$, we do not observe any accepted point mutation.

2.4.1 PAM model

Here, we explain the construction of the PAM model from real data. Dayhoff et al. ([DSO78]) were the first to propose an amino-acid model, based on tabulating changes among closely related sequences. Since they wanted to avoid the possibility of having multiple substitutions, they considered clusters of proteins with 85% or more of identical sites. For each cluster, their approach involved a phylogenetic analysis where, rather than comparing two amino-acid residues directly, they compared them to the inferred common ancestor of the sequences in the cluster. To achieve it, they used a phylogenetic reconstruction method called *maximum parsimony*. Maximum parsimony is not going to be described here (see Section 8.1 of [HA05]), but the principle is that the inferred tree should minimize the total number of amino-acid substitutions required (see Figure 2.3).

Also, as the parsimony method does not provide a unique tree, the ambiguous changes were statistically distributed.

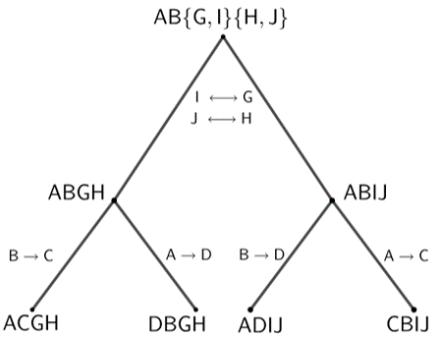


Figure 2.3: Simplified phylogenetic tree inferred by maximum parsimony from an alignment of four amino-acid sequences. Inferred sequences are shown at the interior nodes. Amino-acid exchanges are indicated along the branches. In the sequence at the root, we indicate by brackets those positions where different amino-acids can be chosen.

After inferring the sequence of the common ancestor, one can identify the amino-acid exchanges to obtain a symmetric matrix A, where each entry $A_{ij} = A_{\varsigma_i\varsigma_j}$ is the number of accepted point mutations between ς_i and ς_j (see matrix A in Figure A.2). The matrix A is symmetric because the parsimony method produces unrooted trees. As a consequence, we do not have any information about the direction the changes have occurred. Moreover, time-reversibility is usually assumed, so the same changes in both directions are expected, i.e., $\pi_{\varsigma_i} P(\varsigma_j|\varsigma_i) = \pi_{\varsigma_j} P(\varsigma_i|\varsigma_j)$.

In order to derive the transition matrix $S(t)$, it can be assumed that, for a short time δt, the substitution probabilities are proportional to the substitution rates, i.e., they vary linearly with time. Hence, entries of $S(\delta t)$ can be estimated by:

$$S_{ij} = \lambda \frac{A_{ij}}{n_i} \qquad \text{for } i \neq j$$

where λ is a constant of proportionality to be determined, and n_i is the total number of times that amino-acid ς_i appears in the data. Also, the relative frequency can be estimated as $\pi_i = \dfrac{n_i}{N}$, where $N = \sum\limits_i n_i$ is the total number of amino-acids in the dataset (see Figure A.3).

Dayhoff et al. called *PAM1 matrix* (see Figure A.4) the substitution matrix corresponding to an average of 1% of amino-acid changes. Under this assumption, the parameter λ above can be determined by imposing that the fraction of sites that have changed is

$$\sum_i \pi_i \Big(\sum_{j \neq i} S_{ij} \Big) = \sum_i \sum_{j \neq i} \pi_i \lambda \frac{A_{ij}}{N \pi_i} = \frac{\lambda}{N} \sum_i \sum_{j \neq i} A_{ij} = 0.01.$$

If we define $A_{\text{tot}} := \sum_i \sum_{j \neq i} A_{ij}$ as the total number of exchanges, i.e., the sum of the elements of A, we derive that

$$\lambda = 0.01 \frac{N}{A_{\text{tot}}}.$$

As PAM1 should be a Markov matrix, we complete its diagonal entries as $S_{ii} = 1 - \sum_{j \neq i} S_{ij}$.

Moreover, notice that it defines a time-reversible model:

$$\pi_i S_{ij} = \frac{\lambda}{N} A_{ij} = \frac{\lambda}{N} A_{ji} = \pi_j S_{ji} \quad \text{with } i \neq j.$$

Finally, once PAM1 is defined for δt, PAMk is defined as the substitution matrix for $k\delta t$. So that PAMk = $(\text{PAM1})^k$ (see PAM250 in Figure A.5). Note that for such a matrix

$$1 - \sum_i \pi_i S_{ii}^k$$

is the proportion of amino-acids expected to change.

2.4.2 BLOSUM model

Different models of amino acids are proposed in the literature. The importance of the BLOSUM model comes from the fact that it is obtained by a different approach to that of PAM: Henikoff and Henikoff ([HH92]) proposed a substitution model from ungapped alignments without reconstructing its phylogenetic tree. They defined a symmetric matrix A, whose entries A_{ij} are the number of times each amino-acid is aligned with each other (see Figure 2.4). The frequencies π_i were obtained as in the PAM matrices.

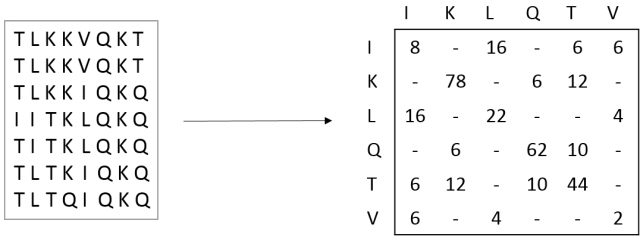


Figure 2.4: An alignment of 7 species (on the left) to obtain a symmetric matrix A (on the right). Its entries are the number of times the pairs of aligned amino-acids appeared in the data.

The method by which matrix A is constructed is understood following the example of Figure 2.4. In case of the first column, as we have 7 sequences and for each amino-acid we can make 6 possible pairs in the column, we have $7 \cdot 6 = 42$ possible amino-acid pairs. Among those, we observe $6 \cdot 5 = 30$ to A_{TT}, 6 to A_{TI} and 6 to A_{IT}. The matrix A is obtained following the same methodology for the rest of the columns of the alignment.

Then, they calculated the fraction of aligned pairs as $n_{ij} = \dfrac{A_{ij}}{A_{\text{tot}}}$ to obtain the estimated probabilities

$$S_{ij} = \frac{n_{ij}}{\pi_i} = \frac{A_{ij}/A_{\text{tot}}}{n_i/N} = \lambda \frac{A_{ij}}{n_i} \quad \text{with now } \lambda = \frac{N}{A_{\text{tot}}}.$$

Similar to the case of the PAM matrices, they defined BLOSUMk. Given an alignment, the sequences were grouped into clusters if they have a percentage of identity greater than k. Then, pairs in a cluster were not considered and pairs between clusters were weighted as a single sequence.

In fact, while in [DSO78] the aim was to develop an evolutionary model and to perform a weight matrix for sequence alignments, BLOSUM was designed mainly for sequence alignments. For this reason, BLOSUMk does not reflect a realistic transition matrix for amino-acids because, when sequences were grouped in clusters, the authors did not consider multiple substitutions. Then, while k is higher, BLOSUMk matrices reflect worst the real accepted point mutations.
Consequently, a matrix calculated by BLOSUM method counts pairs of aligned amino-acids, while a PAM matrix counts the number of substitutions. All this together leads to the reason why BLOSUM matrices have not been widely used in phylogenetic reconstruction.

2.4.3 Other models

When PAM matrices were proposed, relatively few protein sequences were available. Since then, several groups have used much larger datasets to derive improved substitution models. Their methodology differ in obtaining the matrix A, but then they follow the same strategy as the PAM model. Here we present some of the most popular models:

(1) JTT model ([JTT92]) was performed estimating the phylogenetic tree by a distance reconstruction method[1] ([FM67]) for each protein family in the database. Then, to build the matrix A, for each phylogeny they selected the pair of sequences, which were nearest among those having 85% of identity or more, to count the differences between them. This pair of sequences was then discarded to avoid recounting changes and to be able to repeat the methodology with a new pair until these pairs of sequences in all protein families are left.

(2) DCMut and JTTDCMut models ([KG05]) are new versions of the PAM and JTT model, which were calculated obtaining their rate matrix Q from the empirical data, by the following approach

$$q_{ij} = \frac{A_{ij}}{n_i} \quad \text{for } i \neq j.$$

[1] Assuming an evolutionary model, we have the corresponding evolutionary distance. Then a distance reconstruction method gives rise to a distance for each pair of sequences, obtaining a matrix distance, which it is used for a clustering algorithm to reconstruct the topology.

They showed some improvements in getting the rate matrix and, from here, the substitution matrix. Moreover, they manifested that eigen-decomposition method (see Remark 2.1.11) has convergence problems.

(3) WAG model ([WG01]) was performed using maximum likelihood (ML) as the reconstruction method (see Chapter 3.1).
To reduce computational time, their approach consisted of three parts: First, the topology of each family was reconstructed by a distance method. Then, assuming that this topology was correct, its branch lengths were estimated by ML under the JTT+F model[2]. Finally, they fix these branch lengths modulus one parameter to reconstruct the topology, which was used to perform the analogous matrix A of the PAM model.

(4) LG model ([LG08]) used the same process as WAG, but they incorporated the variability of evolutionary rates across sites ($+\Gamma$) in the matrix estimation. Also, they used a much larger database.

All these models are time-reversible.

Moreover, sequence-specific models have been developed, such as cpREV for chloroplasts ([AWMH00]) or FLU for influenza virus ([DLGL10]).

[2] +F indicates that the states distribution of root π^r have been inferred from the data itself and not from the standard model.

3 Reconstruction methods

All methods of phylogenetic reconstruction are based on some kind of character evolution model, either implicitly or explicitly. As every phylogenetic tree represents a hypothesis about the speciation of our taxa, they provide direct knowledge of evolutionary history.

Reconstruction methods can be classified in different ways. If we consider the input data we have *character methods* and *distance methods*. While the former use the aligned sequences directly during tree inference, the latter transform the sequence data into pairwise distances, and then use the matrix during tree building, ignoring characters.

Alternatively, regarding the reconstruction process, *optimization methods* assign particular scores to each of the possible solutions (trees), and the problem is to find the best possible solution. On the contrary, *algorithmic methods* directly construct a tree and, as they do not search in the space tree, so they are much faster. On the other hand, offering a single solution does not allow us to know if there are other trees that are slightly worse than the best.

All of them try to reconstruct the topology of the evolutionary tree. But, also some of them try to infer the branch lengths, the ancestral sequences and/or the root position.

In this chapter, we will introduce two phylogenetic reconstruction methods which will be experimentally compared in the next chapter: Maximum Likelihood (ML) and Singular Value Decomposition (SVD) plus Weight Optimization (WO), denoted as SVD+WO. Both are character based methods, but ML also infers the branch lengths. Moreover, while ML is an optimization method, SVD+WO is classified in the algorithmic methods.

Before describing these two methods, some concepts are introduced:

3.0.1 Definition (Joint distribution) Given a tree $\mathcal{T}$ with leaves $1, 2, \ldots, n$ and an evolutionary model $\mathcal{M} = \{\{S_e\}_{e \in E(\mathcal{T})}, \pi^r\}$, we can construct the vector of the joint random variables of these leaves $X = (X_1, X_2, \ldots, X_n)$, so each column of an alignment is an observation of this vector of random variables. Then, the *joint distribution* $p_{x_1 \ldots x_n}$ is the probability that the leaves take the states $x_1, ..., x_n \in \Sigma$:

$$p_{x_1 \ldots x_n} := P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n).$$

Since the evolutionary processes at different edges are independent and only depend on the common node, the joint distribution $p_{x_1 \dots x_n}$ can be expressed in terms of the parameters that determine the evolutionary model:

$$p_{x_1 \dots x_n} = \sum_{(x_v)_{v \in \mathrm{Int}(\mathcal{T})}} \pi^r_{x_r} \prod_{e \in E(\mathcal{T})} S_e(x_{p(e)}, x_{c(e)}),$$

where the sum is over all possible states $x_v \in \Sigma$ at the interior nodes v, which includes $x_r \in \Sigma$ the state of the root and where $x_{p(e)}, x_{c(e)} \in \Sigma$ are the state of the parent node of the edge e and the state of the descendant node of the edge e, respectively. Notice that the subindex $c(e)$ is i if e is a terminal edge ending at the leaf i.

3.0.2 Remark (Identifiability) [AR03] The root position cannot be identified from the joint distribution, i.e., different root placements and different transition matrices may give rise to the same joint distribution. However, the continuous parameters π^r and $\{S_e\}_{e \in E(\mathcal{T})}$ are identifiable (in general) if one assumes some mild extra conditions (see the reference mentioned above).

3.0.3 Example Consider the tree in Figure 3.1.

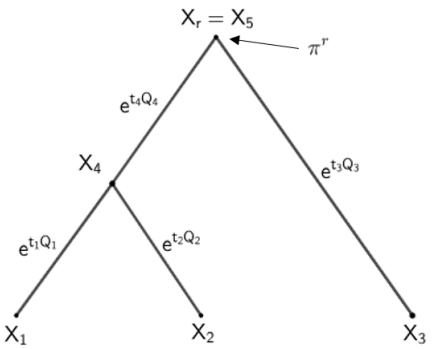


Figure 3.1: An evolutionary model on a rooted tree $\mathcal{T}$ of three leaves. Transition matrices for each evolutionary process are indicated along the branches.

Then $p_{x_1 x_2 x_3} = P(X_1 = x_1, X_2 = x_2, X_3 = x_3)$ is the probability of observing the states $x_1, x_2, x_3 \in \Sigma$ at the leaves, which in terms of the transition matrices of the model $\mathcal{M}$ is

$$p_{x_1 x_2 x_3} = \sum_{x_r, x_4 \in \Sigma} \pi^r_{x_r} S_3(x_r, x_3) S_4(x_r, x_4) S_1(x_4, x_1) S_2(x_4, x_2)$$

where, for example, $S_3(x_r, x_3)$ is the probability that the evolutionary process $X_r \xrightarrow{S_3} X_3$ suffers a mutation from the state x_r to x_3, i.e., the entry of S_3 corresponding to $P(x_3 \mid x_r)$.
Then, for example, if $S_1 = S_2 = S_4 = \mathrm{PAM1}$, $S_3 = \mathrm{PAM250}$ and π^r is the root distribution established by the Dayhoff model (see figures A.3, A.4 and A.5 for the exact values), the joint distribution $p_{\mathtt{VIL}}$ is equal to:

$$p_{\mathtt{VIL}} = \sum_{x_r, x_4 \in \Sigma} \pi^r_{x_r} \text{ PAM250}(x_r, \mathtt{L}) \text{ PAM1}(x_r, x_4) \text{ PAM1}(x_4, \mathtt{V}) \text{ PAM1}(x_4, \mathtt{I}) =$$

$$= \pi^r_{\mathtt{A}} \text{ PAM250}(\mathtt{A}, \mathtt{L}) \text{ PAM1}(\mathtt{A}, \mathtt{A}) \text{ PAM1}(\mathtt{A}, \mathtt{V}) \text{ PAM1}(\mathtt{A}, \mathtt{I}) + \pi^{\mathtt{r}}_{\mathtt{A}} \text{ PAM250}(\mathtt{A}, \mathtt{L}) \text{ PAM1}(\mathtt{A}, \mathtt{R})$$
$$\text{PAM1}(\mathtt{R}, \mathtt{V}) \text{ PAM1}(\mathtt{R}, \mathtt{I}) + ... + \pi^{\mathtt{r}}_{\mathtt{V}} \text{ PAM250}(\mathtt{V}, \mathtt{L}) \text{ PAM1}(\mathtt{V}, \mathtt{V}) \text{ PAM1}(\mathtt{V}, \mathtt{V}) \text{ PAM1}(\mathtt{V}, \mathtt{I}).$$

Now we consider the alignment represented in Table 3.1:

Seq 1	T	L	K	K	V	Q	K	Q
Seq 2	T	L	K	K	I	Q	K	Q
Seq 3	T	I	T	T	L	Q	K	Q

Table 3.1: A multiple sequence alignment D of protein sequences of three sequences.

As we have the assumption that our sequences are independent and identically distributed (i.i.d.), the probability that the evolution in the $\mathcal{T}$ tree of Figure 3.1 has resulted in the alignment D of Table 3.1 is equal to

$$p_{\mathtt{TTT}} * p_{\mathtt{LLI}} * (p_{\mathtt{KKT}})^2 * p_{\mathtt{VIL}} * (p_{\mathtt{QQQ}})^2 * p_{\mathtt{KKK}} \equiv P(D \mid \theta).$$

$P(D \mid \theta)$ is called the *likelihood* that we observe the alignment D under θ, where $\theta = \{\mathcal{T}, \mathcal{M}\}$ are the parameters of the phylogeny.

3.1 Maximum Likelihood

Given a phylogeny $\theta = (\mathcal{T}, \mathcal{M})$ with its tree topology $\mathcal{T}$ and evolutionary parameters $\mathcal{M}$, we can calculate the likelihood that our alignment D would have evolved in these conditions, $\mathcal{L}(\theta) = P(D \mid \theta)$. The maximum likelihood (ML) criterion is to choose the parameters $\hat{\mathcal{M}}$ and the tree $\hat{\mathcal{T}}$, $\hat{\theta} = (\hat{\mathcal{T}}, \hat{\mathcal{M}})$, that maximize this likelihood:

$$P(D \mid \hat{\theta}) = \max_{\theta} P(D \mid \theta).$$

Although we have three qualitatively different types of parameters when calculating the corresponding likelihood: the tree topology, the branch lengths and the evolutionary model, we can fix some of them while others are optimized. For example, $\mathcal{L}(\hat{\theta})_{|\mathcal{T}_0}$ will denote the ML restricted to $\mathcal{T}_0$.

In practice, considering the logarithm makes the computation simpler, thanks to the fact that the logarithm transforms products in sums and is a strictly increasing function (so maxima are reached at the same points). For example, if we take the Example 3.0.2, where we have the alignment D of the Table 3.1, then

$$\ln \mathcal{L}(\theta) = \ln P(D \mid \theta) = \ln p_{\mathtt{TTT}} + \ln p_{\mathtt{LLI}} + 2 \ln p_{\mathtt{KKT}} + \ln p_{\mathtt{VIL}} + 2 \ln p_{\mathtt{QQQ}} + \ln p_{\mathtt{KKK}}.$$

As the number of tree topologies grows factorially on the number of taxa, the problem of finding the maximum likelihood phylogeny is faced with heuristic algorithms, instead of doing an exhaustive search in the tree space. Moreover, these algorithms need to combine continuous and discrete optimization, the former for branch lengths or the entries of the transition matrices on a fixed tree and the latter for topology trees. In addition, numerical optimization methods can converge to local maxima.

A local search of ML algorithm (see Algorithm 3.1) starts with an initial tree $\mathcal{T}_0$ (usually obtained by a distance method), to then compute its maximum likelihood restricted to its topology: $\ln \mathcal{L}(\hat{\theta})_{|\mathcal{T}_0}$. This calculation is accomplished by applying numerical routines like Newton–Raphson or Brent's method ([PFTV92]).
Then, it obtains the neighboring trees (we will call it $\mathcal{N}(\mathcal{T}_0)$), which are minor modifications of that tree using rearrangement operations (see [SOWH96]), and computes this restricted maximum likelihood for each tree until it finds an improved likelihood. Next, it switches to that tree and restarts the procedure. The process is repeated until a tree does not have any neighbors with higher likelihood. Although there are a finite number of possible trees, note that there is no guarantee that the algorithm finishes in a reasonable time, because the number of tree topologies grows exponentially with the leaves considered.

Algorithm 3.1 Maximum likelihood

Input: An alignment D of n taxa and a partition $\bigcup_{i=1}^{r} D_i$ indicating an r-mixture
Output: A phylogeny of this n taxa

Take an initial $\mathcal{T}_0$ by a distance method
procedure `findImprovedLikelihood`($\mathcal{T}_0$)
  Compute $\hat{\theta} = \arg\max \ln \mathcal{L}(\theta)_{|\mathcal{T}_0}$ by Newton–Raphson method (starting in a random θ_0)
  $\mathcal{T}_F = \mathcal{T}_0$ and $\theta_F = \hat{\theta}$
  Obtain $\mathcal{N}(\mathcal{T}_0)$ by a rearrangement operation
  for each $\mathcal{T}_i \in \mathcal{N}(\mathcal{T}_0)$ **do**
    Compute $\hat{\theta} = \arg\max \ln \mathcal{L}(\theta)_{|\mathcal{T}_0}$ by Newton–Raphson method (starting in a random θ_0)
    if $\ln \mathcal{L}(\hat{\theta})_{|\mathcal{T}_i} > \ln \mathcal{L}(\hat{\theta})_{|\mathcal{T}_0}$ **then**
      $\mathcal{T}_F$=`findImprovedLikelihood(`$\mathcal{T}_i$`)`
      Stop for
    end if
  end for
  return $(\mathcal{T}_F, \theta_F)$
end procedure

To treat r-mixtures, the weighted log-likelihood is computed, i.e., if l_i is the length of the alignment D_i, then $\ln P(D \mid \theta) = l_1 P(D_1 \mid \theta) + ... + l_r P(D_r \mid \theta)$.

Generally, in order to make the optimization much more tractable, the same rate matrix is considered in all edges (homogeneous across lineages). Furthermore, time-reversibility is necessary for most of ML algorithms, such that any node can be considered as the root in the calculation of likelihood, as there is no time-direction.

In addition, ML estimators have important statistical properties as well. For instance, it is possible to prove in great generality that maximum likelihood estimators are *statistically consistent*. This means that if our data is generated according to certain phylogenetic parameters, then as the length of the alignment is increased to infinity, the estimators converge to the true parameters. ML estimators are also *asymptotically efficient*, in that they have minimal variance as the length of the alignment is increased ([AR05]).

3.1.1 Remark (ModelFinder) A usual strategy of software packages which reconstruct phylogenies using the ML method is to determine first the state distribution π^r and the rate matrices $\{Q_e\}_{e\in E(\mathcal{T})}$ of the evolutionary model $\mathcal{M}$ to fix them, and then compute the optimal branch lengths and the optimal tree. Thus, there will be less parameters in the optimization. Recall that the ML method assumes homogeneity across lineages, then there will be the same rate matrix for all edges in $\mathcal{T}$.

To achieve it, a statistical criterion for model selection is used (it will not be explained here, for more information see [AZP05]). One example is the software IQ-TREE ([NSHM15]), which uses *ModelFinder* to determine the best-fit model.

In the case of nucleotide models, ModelFinder studies all the possible rate matrices. On the other hand, for amino-acid models, it only studies the available empirical models (see Table B.2) because there would be too many parameters to determine with this alphabet of 20 characters.

To be able to apply this strategy to an r-mixture, the evolutionary model of each partition is determined independently to the others, specifying in which partition each site belongs to. Then, it is evaluated whether some partitions can be merged in the same evolutionary model, without affecting the criterion for model selection. This option in IQ-TREE is called *ModelFinder+Merge*. On the other hand, if it does not try to merge the evolutionary models, the option is ModelFinder as before, but now specifying the partitions.

It is worth noting that previously to apply the option Merge, ModelFinder always treats each partition independently to the others and considers the possibility that the partition has evolved under a heterogeneous model across sites (while for the reconstruction methods that are presented below, one partition means that it has evolved under a homogeneous model across sites).

3.2 Singular Value Decomposition

This section follows the results presented in [Cas12] and [CFS10]. Given the topology $\mathcal{T}$ of an evolutionary tree of n-taxa and an evolutionary model $\mathcal{M} = \{\{S_e\}_{e\in E(\mathcal{T})}, \pi^r\}$, we get back to the joint distribution expression:

$$p_{x_1\dots x_n} = \sum_{(x_v)_{v\in \mathrm{Int}(\mathcal{T})}} \pi^r_{x_r} \prod_{e\in E(\mathcal{T})} S_e(x_{p(e)}, x_{c(e)}),$$

So, the joint distribution can be expressed as a polynomial function in the parameters of $\mathcal{M}$.

Then, for an evolutionary model with d free parameters on the tree topology $\mathcal{T}$ and given a general alphabet of $\Sigma = \{\varsigma_1, ..., \varsigma_k\}$ with k-states, we can associate to the tree a polynomial map

$$\begin{array}{rcl} \varphi_{\mathcal{T}} : \mathbb{R}^d & \longrightarrow & \mathbb{R}^{k^n} \\ \{\{S_e\}_{e\in E(\mathcal{T})}, \pi^r\} & \mapsto & p^{\mathcal{T}} = (p_{\varsigma_1\ldots\varsigma_1}, p_{\varsigma_1\ldots\varsigma_2}, p_{\varsigma_1\ldots\varsigma_3}, \ldots, p_{\varsigma_k\ldots\varsigma_k}), \end{array}$$

which maps any d-tuple of parameters to the joint distribution $(p_{x_1\ldots x_n})_{x_1,\ldots,x_n\in\Sigma}$ of the k^n possible observations at the leaves.

3.2.1 Example We consider a rooted tree $\mathcal{T}$ on three species. If we take GMM for proteins, then we have $4{\cdot}380 = 1520$ parameters, due to the substitution matrices S, and 19 parameters, due to π^r (as S are a Markov matrices and the components sum of π^r is 1). Then it has the following polynomial map:

$$\begin{array}{rcl} \varphi_{\mathcal{T}} : \mathbb{R}^{1539} & \longrightarrow & \mathbb{R}^{20^3} \\ \{\{S_e\}_{e\in E(\mathcal{T})}, \pi^r\} & \mapsto & p^{\mathcal{T}} = (p_{\varsigma_1\varsigma_1\varsigma_1}, p_{\varsigma_1\varsigma_1\varsigma_2}, p_{\varsigma_1\varsigma_1\varsigma_3}, \ldots, p_{\varsigma_{20}\varsigma_{20}\varsigma_{20}}). \end{array}$$

To be able to use algebraic geometry tools, although we work with probabilities, we extend the polynomial map to $\mathbb{C}^{k^n}$. To this end, we remember some concepts that are also presented in [PS05].

3.2.2 Definition (Algebraic variety) An *algebraic variety* V in $\mathbb{C}^n$ is defined as the set of points satisfying a system of polynomial equations $f_i(x_1, ..., x_n) = 0$, with $i = \{1, ..., r\}$:

$$V = \{p \in \mathbb{C}^n \mid f_1(p) = 0, ..., f_r(p) = 0\}.$$

3.2.3 Lemma Given any subset $S \subseteq \mathbb{C}^n$, the set of polynomials vanishing on all the points in S forms an ideal $I(S)$ in $\mathbb{C}[x_1, ..., x_n]$ called the *ideal of S*.

3.2.4 Theorem (Hilbert's Basis Theorem) Every ideal $I \subseteq \mathbb{C}[x_1, ..., x_n]$ can be generated by a finite set of polynomials $f_1, ..., f_m$.

Back to the topic, we denote by $\operatorname{Im}\varphi_{\mathcal{T}}(\mathcal{M})$ the image of $\varphi_{\mathcal{T}}$, which contains all the possible joint distributions that could be generated by some set of parameters in the model $\mathcal{M}$ on the tree topology $\mathcal{T}$.
As, in general, $\operatorname{Im}\varphi_{\mathcal{T}}(\mathcal{M})$ itself is not an algebraic variety, we introduce the next definition:

3.2.5 Definition (Phylogenetic variety) The phylogenetic variety $V_{\mathcal{M}}(\mathcal{T})$ is the smallest algebraic variety containing $\operatorname{Im}\varphi_{\mathcal{T}}(\mathcal{M})$.

3.2.6 Remark As the set of algebraic varieties in $\mathbb{C}^n$ form the closed sets of the Zariski topology, $V_{\mathcal{M}}(\mathcal{T})$ is the Zariski closure of $\operatorname{Im}\varphi_{\mathcal{T}}(\mathcal{M})$. Moreover, $V_{\mathcal{M}}(\mathcal{T})$ is formed by the union of $\operatorname{Im}\varphi_{\mathcal{T}}(\mathcal{M})$ and some algebraic varieties of smaller dimension.

With this in mind, we are interested in the ideal $I(\mathrm{Im}\,\varphi_{\mathcal{T}}(\mathcal{M}))$ or $I_{\mathcal{M}}(\mathcal{T})$, as we will denote. It is known that the ideal $I_{\mathcal{M}}(\mathcal{T})$ coincides with the ideal of the variety $V_{\mathcal{M}}(\mathcal{T})$.

Then every polynomial $f \in I_{\mathcal{M}}(\mathcal{T})$ describes a relation between the theoretical probabilities $p_{x_1\ldots x_n}$, so it has the following structure:

$$f = a + \sum_{\{x_1,\ldots,x_n\}\in\Sigma^n} \sum_{k_{x_1\ldots x_n}=1}^{n_{x_1\ldots x_n}} a_{k_{x_1\ldots x_n}} \left(p_{x_1\ldots x_n}\right)^{k_{x_1\ldots x_n}}$$

where the sum is finite, $a \in \mathbb{C}$ is the constant term, $a_{k_{x_1\ldots x_n}} \in \mathbb{C}$ and $k_{x_1\ldots x_n} \in \mathbb{Z}_{\geq 0}$.

3.2.7 Definition (Invariants and Phylogenetic invariants) With the same notation as above, a polynomial $f \in I_{\mathcal{M}}(\mathcal{T})$ is called an *invariant* of $\mathcal{T}$.
If f is a polynomial that belongs to $I_{\mathcal{M}}(\mathcal{T})$ and that does not belong to $I_{\mathcal{M}}(\mathcal{T}')$ for any other tree topology $\mathcal{T}' \neq \mathcal{T}$ on n leaves, then f is called a *phylogenetic invariant* of $\mathcal{T}$.

An invariant of $\mathcal{T}$ is a polynomial in the variables $(p_{x_1\ldots x_n})_{x_1,\ldots,x_n\in\Sigma}$ that vanishes when $p_{x_1\ldots x_n}$ is the probability of $x_1, \ldots, x_n$ on $\mathcal{T}$ for certain parameters, i.e., $\varphi_{\mathcal{T}}(\mathcal{M}) = (p_{\varsigma_1\ldots\varsigma_1}, \ldots, p_{\varsigma_k\ldots\varsigma_k})$.

So, in order to determine if our tree has evolved under the topology of $\mathcal{T}$ (and not worry about the substitution parameters or branch lengths), the idea is to estimate from our data D the joint probabilities $(p_{x_1\ldots x_n})_{x_1,\ldots,x_n\in\Sigma}$ and see if they are zeros of the phylogenetic invariants of the topology $\mathcal{T}$. Then, the next goal is to find phylogenetic invariants which can help us to recover the topology of the tree.

3.2.8 Definition (Bipartition) Given a set L with two non empty subsets $A, B \subset X$, we say that A and B are a *bipartition* of X if $X = A \cup B$ and $A \cap B = \emptyset$. We will write $A \mid B$.

If $\mathcal{T}$ is an unrooted trivalent tree whose leaves are labelled by the set L then each edge in $\mathcal{T}$ induces a bipartition on L, corresponding to the two subsets of leaves split by that edge.

3.2.9 Definition (Split) A *split* $\{A, B\}$ in a tree is a bipartition of the leaves obtained by removing an edge of the tree.

3.2.10 Definition (Flattening matrix) Let $A \mid B$ be a bipartition of the leaves of a tree $\mathcal{T}$ and let $\tilde{X}_A$ and $\tilde{X}_B$ be the random variables associated to the leaves of A and B, respectively. Then $\tilde{X}_A$ and $\tilde{X}_B$ can take $a = k^{|A|}$ and $b = k^{|B|}$ states respectively. Given a vector $p^{\mathcal{T}} = (p_{x_1\ldots x_n})_{x_1,\ldots,x_n\in\Sigma} \in \mathbb{C}^{k^n}$ we define the *flattening* $Flatt_{A|B}(p^{\mathcal{T}})$ as the $a \times b$ matrix whose entries are the joint distributions of the observations of $\tilde{X}_A$ and $\tilde{X}_B$:

$$Flatt_{A|B}(p^{\mathcal{T}}) = \begin{matrix} \\ \text{States of } \tilde{X}_A \end{matrix} \begin{matrix} \text{States of } \tilde{X}_B \\ \begin{pmatrix} p_{u_1v_1} & p_{u_1v_2} & \cdots & p_{u_1v_b} \\ p_{u_2v_1} & p_{u_2v_2} & \cdots & p_{u_2v_b} \\ \vdots & \vdots & \ddots & \vdots \\ p_{u_av_1} & p_{u_av_2} & \cdots & p_{u_av_b} \end{pmatrix} \end{matrix}$$

where the subscripts of each entry are $u_i = \varsigma_{i_1}\varsigma_{i_2}...\varsigma_{i_a}$ and $v_j = \varsigma_{j_1}\varsigma_{j_2}...\varsigma_{j_a}$.

3.2.11 Example Let $\mathcal{T}$ be a tree with 4 leaves. Then, the $Flatt_{12|34}(p^{\mathcal{T}})$ is the 16×16 matrix:

$$Flatt_{12|34}(p^{\mathcal{T}}) = \begin{matrix} \\ \text{States at leaves 1 and 2} \end{matrix} \begin{matrix} \text{States at leaves 3 and 4} \\ \begin{pmatrix} p_{\texttt{AAAA}} & p_{\texttt{AAAR}} & p_{\texttt{AAAN}} & \cdots & p_{\texttt{AAVV}} \\ p_{\texttt{ARAA}} & p_{\texttt{ARAR}} & p_{\texttt{ARAN}} & \cdots & p_{\texttt{ARVV}} \\ p_{\texttt{ANAA}} & p_{\texttt{ANAR}} & p_{\texttt{ANAN}} & \cdots & p_{\texttt{ANVV}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{\texttt{VVAA}} & p_{\texttt{VVAR}} & p_{\texttt{VVAN}} & \cdots & p_{\texttt{VVVV}} \end{pmatrix} \end{matrix}.$$

3.2.12 Theorem (Allman-Rhodes) [AR08] Given a split of an unrooted tree $\mathcal{T}$ such that we have the corresponding bipartition of the leaves $A \mid B$, with $\alpha = |A| \geq 2$ and $\beta = |B| \geq 2$, then $Flatt_{A|B}(p^{\mathcal{T}})$ has rank $\leq k$. Moreover, the $Flatt_{A|B}(p^{\mathcal{T}})$ of the other bipartitions with $|A| = \alpha$ and $|B| = \beta$ has rank $\geq k^2$ for general $p^{\mathcal{T}}$.

Equivalently, all $(k+1) \times (k+1)$ minors of $Flatt_{A|B}(p^{\mathcal{T}})$ are phylogenetic invariants of $\mathcal{T}$.

Although Allman-Rhodes's theorem is the basic statement to develop our phylogenetic reconstruction method, we need to check that we can recover our topology knowing its splits or bipartitions.

3.2.13 Definition (Compatible bipartitions) Two bipartitions $A_1 \mid B_1$ and $A_2 \mid B_2$ of the same set are said to be *compatible* if at least one of the four intersections $A_1 \cap A_2$, $A_1 \cap B_2$, $B_1 \cap A_2$, $B_1 \cap B_2$ is empty.

3.2.14 Proposition The set $bi(\mathcal{T})$ of the $2n-3$ bipartitions induced by the edges of $\mathcal{T}$ is composed of pairwise compatible bipartitions.

3.2.15 Theorem (Buneman) [Bun71] Let L be a set of n elements and S a collection of $2n-3$ bipartitions of L. Then S is formed by pairwise compatible bipartitions if and only if there exists an unrooted tree $\mathcal{T}$ with leaves labelled on L such that $bi(T) = S$. In this case, the tree $\mathcal{T}$ is unique.

Thereby, given an alignment D of n species, we can estimate the joint probability $p^{\mathcal{T}}_{x_1...x_n}$ with the relative frequency of the n-tuple $x_1, ..., x_n$ occurring as a column of the alignment, which we will denoted by $(\rho_{x_1...x_n})_{x_1,...,x_n \in \Sigma}$. Then determine if each possible $A \mid B$ is a split of T by evaluating the rank of the associated flattening matrix.

In practice, our sequences may not have evolved under a mathematical evolutionary model nor on a phylogenetic tree and, as the sequences are finite, the relative frequencies $\rho_{x_1 \dots x_n}$ do not match exactly with the joint probabilities.
Even so, if the evolutionary model considered fits the data well and the length of the alignment is acceptable then phylogenetic invariants of the "correct" tree topology evaluated at the vector of relative frequencies $\rho_{x_1 \dots x_n}$ of columns of the alignment should be close to zero.

Thus, to establish which is the "correct" split, we need to evaluate which of the flattening matrices is the nearest to the matrices of rank less or equal to k. So we will use the distances presented in the Chapter 1.2, more precisely, the Frobenius-distance (see Theorem 1.2.12).

Furthermore, for an unrooted tree with n leaves, for each $k = \{2, ..., \lfloor \frac{x}{2} \rfloor\}$, we would have to evaluate the $\binom{n}{k}$ possible splits $\{A, B\}$ with $|A| = k$ and $|B| = n - k$, so it is non-viable to evaluate each possible split to infer the tree topology. For this reason, we introduce Algorithm 3.2 as a methodology to determine the correct topology of a quartet tree (tree with four leaves), and then we will describe a method (Weight Optimization) to combine quartets to output a tree.

We will call this algorithm SVD because it uses the singular values of the flattening matrix to determine the Frobenius-distance. In the next section, we will use this algorithm to reconstruct trees with a higher number of leaves. As in the case of Algorithm 3.1, SVD is also statistically consistent ([CFS16]).

Algorithm 3.2 SVD

Input: An alignment D of 4 taxa and $r \in \mathbb{N}$ indicating an r-mixture
Output: A topology of this 4 taxa

Compute the vector of absolute frequencies $(\rho_{x_1 \dots x_4})_{x_1,\dots,x_4 \in \Sigma}$ and set $k = |\Sigma|$
$\mathcal{T}_F = 12 \mid 34$ and $finalDist = \infty$
for each $A \mid B \in \{12 \mid 34, 13 \mid 24, 14 \mid 23\}$ **do**
  Compute $Flatt_{A|B}(\rho)$
  Factorize $Flatt_{A|B}(\rho) = UDV^t$ in SVD
  $d_F(Flatt_{A|B}(\rho), E_{rk}) = \sqrt{\sum_{i=rk+1}^{k^2} \sigma_i^2}$
  if $d_F(Flatt_{A|B}(\rho), E_{rk}) < finalDist$ **then**
    $\mathcal{T}_F = A \mid B$ and $finalDist = d_F(Flatt_{A|B}(\rho), E_{rk})$
  end if
end for
return $\mathcal{T}_F$

As a consequence of the following theorem, the algorithm considers the possibility of working with an *r-mixture* with $r < k$:

3.2.16 Theorem Given an alignment D that evolves under $r < k$ different evolutionary models and under the same tree $\mathcal{T}$, if $A \mid B$ is a split of $\mathcal{T}$, then

$$rank(Flatt_{A|B}(p^{\mathcal{T}})) \leq rk.$$

Proof. First, our alignment D can be divided into r alignments, i.e., $D = D_1 \cup D_2 \cup ... \cup D_r$ such that $D_i \cap D_j = \emptyset \quad \forall i, j \in \{1, ..., r\}$. If we denote l_i the length of the alignment D_i and l the length of the whole alignment, then we have:

$$l\ Flatt_{A|B}(p^{\mathcal{T}}) = l_1 Flatt_{A|B}(p^{\mathcal{T}}_{\mathcal{M}_1}) + l_2 Flatt_{A|B}(p^{\mathcal{T}}_{\mathcal{M}_2}) + ... + l_r Flatt_{A|B}(p^{\mathcal{T}}_{\mathcal{M}_r})$$

So, as $rank(M_1 + M_2) \leq rank(M_1) + rank(M_2)$ and $rank(kM_1) = rank(M_1)$ with $k \in \mathbb{Z}$, we have the following inequality:

$$rank(Flatt_{A|B}(p^{\mathcal{T}})) = rank(l\ Flatt_{A|B}(p^{\mathcal{T}})) \leq$$
$$\leq rank(l_1 Flatt_{A|B}(p^{\mathcal{T}}_{\mathcal{M}_1})) + ... + rank(l_r Flatt_{A|B}(p^{\mathcal{T}}_{\mathcal{M}_r})) \leq k + ... + k = rk.$$

□

In this case, we only need to compute the Frobenius-distance of the general flattening matrix to the matrices with the rank less or equal to rk, without the need to know which are the partitions (unlike ML). Note that we need a strict inequality $rk < min\{k^{|A|}, k^{|B|}\}$ to be able to differentiate if $A \mid B$ is a split of $\mathcal{T}$, so, for example, with amino-acid alphabet and quartet trees we can consider up to 19 partitions, as their flattening matrices size are $k^2 \times k^2$.

3.3 SVD + Weight Optimization

As pointed out above, in order to face the impossibility to evaluate each split, it is necessary to use other types of strategies. While most ML software use heuristic approaches, there exist alternatives as *super-tree methods*, which combine the set of inferred smaller trees obtained by some reconstruction method into a resulting tree that attempts to be consistent with the greatest number of these smaller trees. In particular, *quartet-based methods* construct a tree from all possible *quartets* (trees with four leaves) as smaller trees.

Thus, quartet-based methods try to provide the topology of a tree consistent with the greatest number of quartet topologies.

From now on, we will adopt the notation 12 | 34, 13 | 24 and 14 | 23 for the three possible quartets (as in Figure 1.5). Also, we define a *4-tuple* of $\mathcal{T}$ as a subset of four leaves.

3.3.1 Definition (Satisfy $\mathcal{T}_2$) For two trees $\mathcal{T}_1$ and $\mathcal{T}_2$, we say that $\mathcal{T}_1$ *satisfies* $\mathcal{T}_2$ if the set of leaves $\mathcal{L}(\mathcal{T}_2) \subseteq \mathcal{L}(\mathcal{T}_1)$ and $\mathcal{T}_{1|\mathcal{L}(\mathcal{T}_2)} = \mathcal{T}_2$, i.e., $\mathcal{T}_2$ can be obtained from $\mathcal{T}_1$ by removing the appropriate set of edges.

3.3.2 Definition (Consistency) For a set of trees $X = \{\mathcal{T}_1, ..., \mathcal{T}_t\}$ with possibly overlapping leaves, we say that a tree $\mathcal{T}^*$ is *consistent* with X if there exists a tree $\mathcal{T}^*$ over the union set of leaves of the trees in X that satisfies every tree $\mathcal{T}_i \in X$.

The problem of finding such a consistent tree $\mathcal{T}^*$, or a close one if none exists, is known as the supertree problem.

Given our alignment D, the goal of quartet-based methods is to find the tree that is consistent with the maximum number of the $\binom{n}{4}$ quartets. To achieve this goal, quartets are inferred by a reconstruction method which outputs also a reliability measure of every possible quartet.

More precisely, this reliability is given by means of a system of weights. For SVD, it assigns to each of the three possible tree topologies of each 4-tuple a weight which takes into account Allman-Rhodes's theorem and such that the sum of weights is one.

To this end, one computes the Frobenius-distance for the three possible flattening matrices, calculates its inverses and normalizes so that the sum is one.
In summary, if we have an r-mixture and we denote each possible topology $\mathcal{T}_1 = 12 \mid 34, \mathcal{T}_2 = 13 \mid 24$ and $\mathcal{T}_3 = 14 \mid 23$ of a 4-tuple $\{1, 2, 3, 4\}$, then the weights $w(\mathcal{T}_i)$ are equal to:

$$w(\mathcal{T}_i) = \frac{d_F(Flatt_{\mathcal{T}_i}(\rho), E_{rk})^{-1}}{\sum_j d_F(Flatt_{\mathcal{T}_j}(\rho), E_{rk})^{-1}} \quad \text{for } i \in \{1, 2, 3\}, \tag{3.1}$$

which can be computed more easily as:

$$w(\mathcal{T}_i) = \frac{\delta(\mathcal{T}_i)}{\delta(\mathcal{T}_1) + \delta(\mathcal{T}_2) + \delta(\mathcal{T}_3)}$$

with $\delta(\mathcal{T}_i) = d_F(Flatt_{\mathcal{T}_j}(\rho), E_{rk})\ d_F(Flatt_{\mathcal{T}_k}(\rho), E_{rk})$ where $\{i, j, k\} = \{1, 2, 3\}$.

We now describe the quartet-based method that will be used: Weight Optimization (WO) [RG01a].

Given a set of taxa $\mathcal{L} = \{1, 2, 3, ..., n\}$ and given weights, denoted by $w(q)$ for each quartet $q = \{ij \mid kl,\ ik \mid jl,\ il \mid jk\}$ of each 4-tuple $\{i, j, k, l\}$ in $\mathcal{L}$, WO starts with a random 4-tuple and selects the topology with highest weight. Then, at each step, it adds one of the remaining taxa until a complete n-tree is obtained. Every step tries to provide the safety option such that the new tree $\mathcal{T}$ is consistent with the previous one and also tries to maximize a weight criterion $W(\mathcal{T})$:

$$W(\mathcal{T}) = \sum_{q \in \mathcal{Q}_\mathcal{T}} w(q)$$

where q denotes the topology of a quartet tree, w its weight from the SVD algorithm and $\mathcal{Q}_\mathcal{T}$ is the set of quartets induced by $\mathcal{T}$.

Hence, as at each step it tries to maximize a criterion, but does not guarantee finishing in the optimal tree, WO is a greedy algorithm.

In more detail, once WO has finished the step $k-1$ and is going to begin with step k, the algorithm has already built a tree $\mathcal{T}_{k-1}$ with the taxa $\mathcal{L}(\mathcal{T}_{k-1}) = \{x_1, ..., x_{(k-1)+4}\}$

$\subset \mathcal{L}$. To choose the next taxon $i \in \mathcal{L}\backslash\mathcal{L}(\mathcal{T}_{k-1})$, it repeats the following process for each i:

It initializes the scores of all edges to 0, then, it considers all quartets with leaves x, y, z and i such that $x, y, z \in \mathcal{L}(\mathcal{T}_{k-1})$.

Given x, y, z three different nodes of a tree $\mathcal{T}$, then there is a single internal node which belongs to the paths (x, y), (x, z) and (y, z). This internal node is called the *median* of x, y, z.

The median of x, y, z divides the tree in three disjoint subtrees $\mathcal{T}_x, \mathcal{T}_y, \mathcal{T}_z$ such that $x \in \mathcal{T}_x, y \in \mathcal{T}_y, z \in \mathcal{T}_z$ (see Figure 3.2). For each quartet q, the method adds $w(q)$ to all edges of $\mathcal{T}_x$ (resp. of $\mathcal{T}_y, \mathcal{T}_z$) if $q = yz \mid xi$ (resp. $q = xz \mid yi$, $q = xy \mid zi$).

Note that the median of x, y, z has degree three, because is an internal node of an unrooted tree. Then, we can consider the median as the "root" of $\mathcal{T}_x, \mathcal{T}_y, \mathcal{T}_z$ such as if the taxon i is added on a edge of T_z, the new tree will satisfy the topology $xy \mid zi$ that we were considering. Thus, all edges in T_z increase their scores by w.

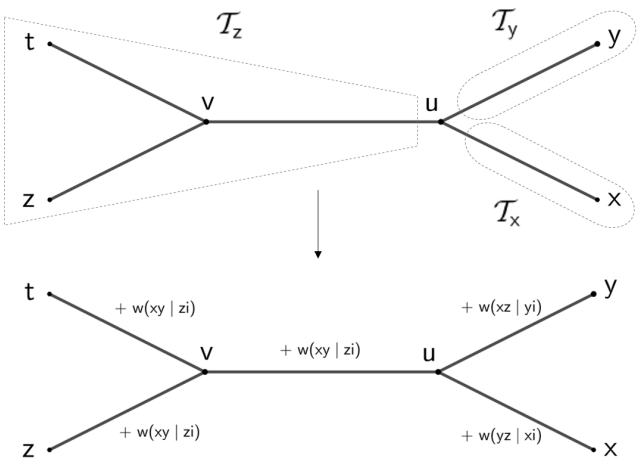


Figure 3.2: The internal node u is the median of x, y, z. It splits the tree into three subtrees $\mathcal{T}_x$, $\mathcal{T}_y$ and $\mathcal{T}_z$. Every edge of $\mathcal{T}$ belongs to a single one of these subtrees. For example, if $e \in \mathcal{T}_z$, then its score is increased by the weight $w(xy \mid zi)$.

When this process has been carried out for each taxon $i \in \mathcal{L}\backslash\mathcal{L}(\mathcal{T}_{k-1})$, we have for each i different scores in the edges of the tree $\mathcal{T}_{k-1}$, denoted by $s_i(e)$ $\forall e \in E(\mathcal{T}_{k-1})$. So, WO has to choose only one of these edges where to attach i, the one that provides the optimal addition.
If we denote e' and e'' the edges such that $s_i(e') \geq s_i(e'') \geq s_i(e)$ $\forall e \in E(\mathcal{T}_{k-1})\backslash\{e', e''\}$, i.e., e and e'' are the edges with the highest score and the second highest score for the taxon i, then the safety $S(i)$ of the taxon i is:

$$S(i) = \frac{s_i(e') - s_i(e'')}{s_i(e') + s_i(e'')}.$$

Note that $s_i(e')$ is the increase of $W(\mathcal{T})$ if the taxon i is added and that $S(i)$ is normalized.

To finish step k, the taxon i_0 with maximum safety $S(i_0)$ is added in the edge e' with maximum score, i.e., in the edge

$$e' = \max_{e \in E_{\mathcal{T}_{k-1}}} s_{i_0}(e). \tag{3.2}$$

3.3.3 Example Figure 3.3 shows how quartets increase the score of the edges, until the taxon is added. Note that in this case, safety is not computed as there is only one possible taxon to be added.

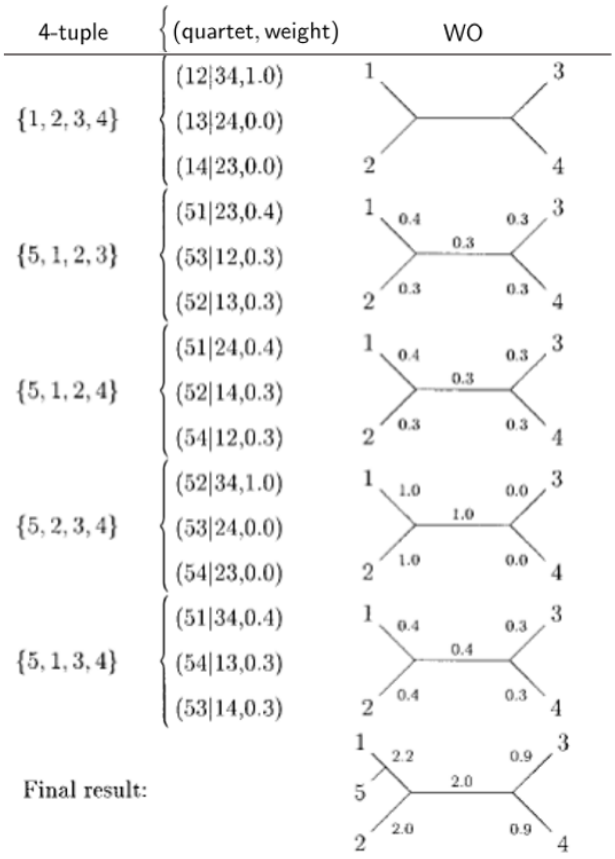


Figure 3.3: A reconstruction of a tree of five leaves using Weight Optimization algorithm ([RG01a]).

Unlike other quartet-based methods, WO fulfills the following property:

3.3.4 Proposition [RG01b] Let $\mathcal{T}$ be a tree and let $Q(\mathcal{T})$ be its set of quartets. If a quartet reconstruction method satisfies that for each 4-tuple of $\mathcal{L}(\mathcal{T})$ the quartet with maximum weight is the one in $Q(\mathcal{T})$ (i.e., the method correctly reconstructs all quartets of $\mathcal{T}$), then the output of WO from this set of weights is $\mathcal{T}$.

Consequently, if the weighting of trees is good enough, WO reconstructs the correct tree independently of the random initial quartet and, therefore, it is not necessary to

compute the majority rule consensus tree. Unfortunately, in our computations we are not in the situation to take advantage of this proposition (Chapter 4), as the weights do not necessarily fulfill this property.

One of the main drawbacks of quartet-based methods is that the reconstructed tree depends on the order in which the quartets are taken into account. To face with this problem, these methods are iterated several times and then they provide a tree $\mathcal{T}^*$ consistent with a majority of the trees obtained.

More precisely, $\mathcal{T}^*$ is called the majority rule consensus tree, because its definition is as follows:

3.3.5 Definition (Majority rule consensus tree) For a set of trees $X = \{\mathcal{T}_1, ..., \mathcal{T}_t\}$, the *majority rule consensus tree* is the tree whose splits appear in more than a half of X.

This tree is unique as the following result shows:

3.3.6 Proposition [MM81] Given a set of trees $X = \{\mathcal{T}_1, ..., \mathcal{T}_t\}$, there exists a unique majority rule consensus tree.

So, our algorithm will reconstruct several times a tree from quartets and then return the majority rule consensus tree, which will be obtained using a python library called DendroPy ([SH10]). More precisely, after obtaining the weights for all quartets induced by each set of data, we have run WO 1000 times and then we have computed the majority rule consensus tree from the set $X = \{\mathcal{T}_1, ..., \mathcal{T}_{1000}\}$.

Next, the reader can find the algorithm 3.3 that combines SVD and WO to reconstruct a tree for $n \geq 4$. Notice that for $n = 4$, we already have the Algorithm 3.2.

Algorithm 3.3 SVD + WO

Input: An alignment D of n taxa, $r \in \mathbb{N}$ indicating an r-mixture and $t \in \mathbb{N}$ the number of replicas of WO that will be computed

Output: A topology of this n taxa

Set $k = |\Sigma|$

for each quartet Q **do**

  for each $A \mid B \in \{12 \mid 34, 13 \mid 24, 14 \mid 23\}$ **do**

    Compute $Flatt_{A|B}(\rho) = UDV^t$

    $d_F(Flatt_{A|B}(\rho), E_{rk}) = \sqrt{\sum_{i=rk+1}^{k^2} \sigma_i^2}$

  end for

  Compute the three weights $w(A \mid B)$ as (3.1)

end for

for each replica $j \leq t$ **do**

  Randomly select four taxa $Q = \{1, 2, 3, 4\}$ ▷ Here WO starts

  $\mathcal{T}_0$ = higher weight of the quartets on Q

  Set $S = \{1, 2, 3, 4\}$ and $R = \{4, ..., n\}$

  while $R \neq \emptyset$ **do**

    for each taxon $i \in R$ **do**

      Reinitialize edge scores to 0

      for each $x, y, z \in S$ **do**

        Get subtrees T_x, T_y, T_z from the median

        Add $w(yz \mid ix)$ to all edges of T_x

        Add $w(xz \mid iy)$ to all edges of T_y

        Add $w(xy \mid iz)$ to all edges of T_z

      end for

      Memorize the best edge for taxon i and its safety $S(i)$

    end for

    $i_0 = \max_i S(i)$

    Add taxon i_0 on the edge e' with maximum score in $\mathcal{T}_{k-1}$ as (3.2)

    Add taxon i_0 to S and remove it from R

  end while ▷ Here WO finishes

  $\mathcal{T}^j = \mathcal{T}_{n-4}$, where $\mathcal{T}_{n-4}$ is the final tree from WO

end for

$\mathcal{T}_F$ = majority rule consensus tree $\{\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^t\}$

return $\mathcal{T}_F$

4 Experimental analysis

This chapter explains the different analyses done and their results in order to measure the potential of the SVD and SVD+WO algorithms. To achieve our main goal, these analyses include the performance of topology reconstruction, but also the computational time needed and a comparison with one of the most widely used ML software: IQ-TREE ([NSHM15]).

To this end, several alignments were simulated and then used for tree reconstruction. For homogeneous across lineages evolutionary models, the alignments were obtained with Seq-Gen software ([RG97]). Otherwise, BppSuite software was used ([G+13]).

As phylogenetic reconstruction methods have significant difficulties with some situations called *long branch attraction*, some of the simulated alignments have been designed in order to examine the behavior of our methods under this situation. Long branch attraction is a phenomenon that occurs when lineages that are separated (i.e., they are not adjacent) and that have evolved with high evolutionary rates (i.e., divergent taxa with very long branches) are erroneously placed as adjacent to the same node of a tree. This may be due, for example, to convergent changes (which are homoplasies) caused by multiple substitutions.

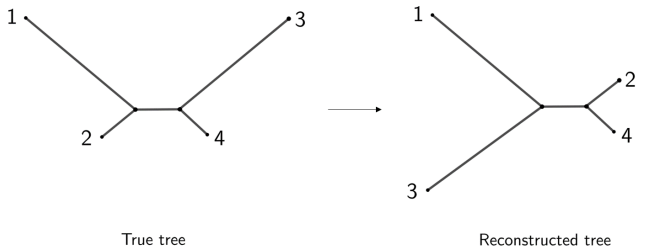


Figure 4.1: Reconstruction methods fail in the phenomenon of long branch attraction, where separated long branches are inferred as adjacent.

All these computations have been performed in the so-called Hercules server, which belongs to the EGB group at the Universitat de Barcelona. Table B.3 shows the computational capability and its structure in clusters.

4.1 Study of SVD and ML on quartets

This section summarizes the study followed after we adapted SVD software to a general alphabet (see the main code in section C.1), implementing it in C++ using the Armadillo Library ([SC16]). We study the performance of SVD and ML on quartets and in the next section we consider larger trees.

First, we detail how the 13500 alignments were simulated. Then, we describe the performed computations to reconstruct the different alignments. Finally, we present the results obtained and a discussion of the advantages and disadvantages of the software used.

4.1.1 Simulated data on quartets

A hundred alignments of four species were generated for each tree (a), each alignment length (b) and each evolutionary model (c) as described:

(a) **Tree**: 9 different unrooted trees were obtained varying the branch lengths of the 12 | 34 topology. More precisely, the internal edge[1] was considered with length 0.1, 0.2 or 0.4 and, for each one of these settings, the terminal edges[2] 1, 2, 3 and 4 were assigned with lengths 0.1, 0.1, 0.1 and 0.1 (type A), or 0.3, 0.1, 0.3 and 0.1 (type B), or 0.3, 0.3, 0.3 and 0.1 (type C), respectively; see Figure 4.2 (the three types of the terminal edges A, B and C correspond to the first, the second and the third row, respectively).

(b) **Alignment length**: 3 different alignment lengths were considered: 1000, 5000 and 10000.

(c) **Model**: 4 different evolutionary models were considered:

- Three models were homogeneous across lineages: JTT, WAG and LG models.
- The fourth model was heterogeneous across lineages. This model had evolved under JTT model on the terminal edges 1 and 2, under WAG model on the internal edge and under LG model on the terminal edges 3 and 4. In order to simulate this heterogeneous model, BppSuite took one random node as the root and then it started with a random root distribution.

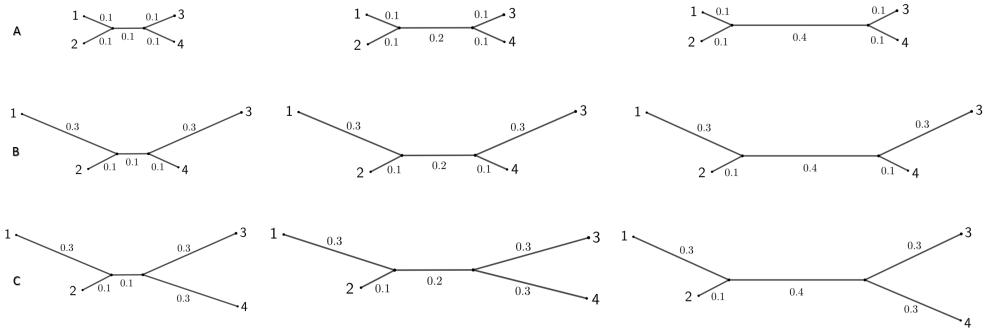


Figure 4.2: The 9 different unrooted trees that were considered in the simulations.

[1] the edge that joins the internal nodes.
[2] the edges that end in the leaves 1, 2, 3 and 4.

In these simulations, the study on the long branch attraction situation was mainly included in the tree with a length 0.1 in the internal edge and with a length 0.3, 0.1, 0.3 and 0.1 in the terminal edges 1, 2, 3 and 4, respectively (type B).

Alignments with mixtures

Moreover, in order to study the performance of the methods on mixture models, the alignments with the same length and on the same tree that were generated under the homogeneous across lineages models were concatenated. Note that the alignments with mixtures had lengths of 3000, 15000 or 30000 and their partitions only differed in the rate matrix considered (JTT, LG, WAG).

Then, a total of 10800 alignments with no mixtures and 2700 alignments with mixtures were simulated. This huge amount of data attempted to cover many distinct situations in the speciation process.

4.1.2 Reconstruction strategies on quartets

We differentiate between alignments with no mixtures and with mixtures. The former was reconstructed following these two strategies:

(1) Using the implemented SVD algorithm (Algorithm 3.2) considering a single partition.

(2) Using the ML algorithm (Algorithm 3.1) of the IQ-TREE software. In the reconstruction process, it first determines the evolutionary model by means of ModelFinder (see Remark 3.1.1).

In the case of the alignments with mixtures, which are the concatenations of three alignments, two strategies for SVD and two for ML were applied in their reconstruction:

(1) Using the implemented SVD algorithm (Algorithm 3.2).

  1.1. Considering only one partition, i.e., $r = 1$.

  1.2. Considering three partitions, i.e., $r = 3$.

(2) Using the ML algorithm (Algorithm 3.1) of the IQ-TREE software.

  2.1. Considering only one partition. Then, ModelFinder is used in the same way as alignments with no mixtures.

  2.2. Considering three partitions. Then, in the reconstruction process, ModelFinder+Merge first determines the best partition scheme by possibly merging partitions (see Remark 3.1.1).

Recall that while SVD (Algorithm 3.2) treats with r-mixtures only specifying the number of partitions, i.e., r is taken as an input parameter, IQ-TREE (Algorithm 3.1) needs to know the precise position of the partitions, i.e., $\bigcup_{i=1}^{r} D_i$ is taken as an input.

4.1.3 Reconstructing quartets: Results and discussion

As above, we differentiate between alignments with no mixtures and with mixtures.

Alignments with no mixtures

Homogeneous models. Beginning with homogeneous models, both SVD and IQ-TREE software performed perfectly for alignments of length 5000 and 10000.

Regarding length 1000, IQ-TREE also performed perfectly. By contrast, SVD failed in a few cases, all of them with trees belonging to the type A. More precisely, for internal edges 0.1 and 0.2 under JTT model (Table 4.1), WAG model (Table 4.2) and LG model (Table 4.3), the results were similar and were slightly better for the internal edge length 0.4. Therefore, it seems that the evolutionary models did not affect the reconstruction performance of SVD.
On the other hand, results show that when the internal edge was longer, SVD performed better. Moreover, terminal edges were also a relevant factor, as only when they were shorter (type A), SVD did not achieve to reconstruct all the replicas perfectly.

We had, then, a slightly better performance of IQ-TREE, but our data did not allow us to measure exactly how significant the difference was.

		Internal edge		
		0.1	0.2	0.4
Types of the terminal edges	A	0.99	0.98	0.99
	B	1	1	1
	C	1	1	1

Table 4.1: Proportion of correctly inferred trees by SVD from the alignments with length 1000 and that had evolved under the homogeneous JTT model.

		Internal edge		
		0.1	0.2	0.4
Types of the terminal edges	A	0.97	0.97	1
	B	1	1	1
	C	1	1	1

Table 4.2: Proportion of correctly inferred trees by SVD from the alignments with length 1000 and that had evolved under the homogeneous WAG model.

		Internal edge		
		0.1	0.2	0.4
Types of the terminal edges	A	0.96	0.98	1
	B	1	1	1
	C	1	1	1

Table 4.3: Proportion of correctly inferred trees by SVD from the alignments with length 1000 and that had evolved under the homogeneous LG model.

Note in particular that both SVD and IQ-TREE perfectly reconstructed all the replicas for the internal edge 0.1 and the type B, which was the case designed to study the long branch attraction situations.

Heterogeneous model. In the case of alignments that had evolved under the heterogeneous model, SVD and IQ-TREE reconstructed correctly all the trees, independently of the length of the alignment and of the branch lengths. Thus we can remark that SVD performed better than in the case of homogeneous models.

It is important to note that although IQ-TREE assumes that the underlying model is homogeneous across lineages, we have seen that it correctly performs on heterogeneous data. This suggests that the three considered models are not so different. This is due to the fact that the different amino-acid models were empirically obtained in a similar manner and always using a general database, so they described similar amino-acid mutations (mainly, following chemical properties).

Alignments with mixtures

Finally, we present the results for the alignments that were concatenated. In this case, also IQ-TREE and SVD inferred correctly all the simulated data, independently if the mixtures were treated as one partition or as three. Results for the minimum length are shown in Table 4.4 to emphasize their importance.

We could conclude that the presence of partitions was not very relevant if they did not differ too much, as in our case: only the rate matrices of the evolutionary models were different, but the branch lengths and the topology were exactly the same in the three partitions.

Notice that as the alignments were concatenated, they had a longer length than when we treated with alignments with no mixtures. This fact influenced a better reconstruction.

			Internal edge		
			0.1	0.2	0.4
Treating like one partition $r = 1$	Types of the terminal edges	A	1	1	1
		B	1	1	1
		C	1	1	1
Treating like three partitions $r = 3$	Types of the terminal edges	A	1	1	1
		B	1	1	1
		C	1	1	1

Table 4.4: Proportion of correctly inferred trees by SVD from the three alignments concatenated. Each partition had a length of 1000 and had evolved under the homogeneous JTT, WAG or LG models. Results are shown as the alignments were considered as one partition and three partitions.

As it has been mentioned, while SVD needs only the number of partitions to specify a mixture (see Algorithm 3.2), IQ-TREE requires knowing to which partition each site belongs (see Algorithm 3.1). Moreover, IQ-TREE does not consider heterogeneous models as SVD. Therefore, despite its excellent results, to face with more complicated data, IQ-TREE needs more information to deal with mixtures than SVD, and only considers the same rate matrix for all the edges. Later, when we present the results obtained for SVD+WO, some of these points are shown more clearly.

4.1.1 Remark The SVD method was developed in the same way as Erik+2 software ([CFS16]). This last software is similar to the SVD algorithm, but it applies a correction to the flattening matrices in order to deal with the long branch attraction situation.

Once Erik+2 has acquired a flattening matrix, it obtains two matrices, the former by a column sum correction and the latter by a row sum correction. These matrices are obtained by dividing any non-zero column (row) by the sum of its entries, making all the non-zero columns (rows) to have the same weight. The Frobenius-distances obtained after normalizing by both rows and columns are taken into account to compute the final score. Both corrections maintain the same rank.

Unfortunately, in general, this methodology can not be applied to an amino-acid alignment. Indeed, we applied the Erik+2 strategy to our data, but the average of quartet correctly inferred was less than one in three, which is a worse result than if the trees had been randomly inferred. A possible reason for this is that the flattening matrices for amino-acids are sparse, due to their great size (as now the alphabet Σ has 20 characters), and then these corrections unbalances the weight of columns (rows) with only one entry with regard to the others. Therefore, we decided not to use this correction of the flattening and use directly SVD.

Running time

When it comes to the computational time issue (under the same conditions: 4 CPUs and the same RAM available), SVD holds a leading position. The average results of running time for one alignment (computed for a sample of 100) are shown depending on the alignment length and the considered partitions (see Table 4.5). The branch lengths and the evolutionary models of the simulated data did not affect in the computational time.

Although a longer length in the alignment should in principle affect the execution time for the SVD software, Table 4.5 does not exhibit this. This is probably because the alignment lengths were not long enough to show a significant difference.

On the other hand, for SVD, the number of partitions should not affect the computational time, because the algorithm only differs in the Frobenius-distance computed.

			Reconstruction strategies	
		Alignment length	SVD	IQ-TREE
No mixtures (one partition)		1000	0.21”	9.54”
		5000	0.24”	23.41”
		10000	0.23”	42.63”
Mixtures	Treating like one partition	3000	0.22”	17.33”
		15000	0.23”	1’ 6.23”
		30000	0.25”	2’ 7.18”
	Treating like three partitions	3000	0.23”	1’ 21.61”
		15000	0.22”	4’ 16.87”
		30000	0.23”	7’ 54.46”

Table 4.5: Average of computational time for SVD and IQ-TREE on one alignment (under the same hardware conditions). Results are shown depending on the alignment length and the considered partitions.

In the case of IQ-TREE, we could see in all settings that there was a clear increase with the alignment length. Moreover, the number of partitions was also a determining factor. In fact, in the case of treating the alignment as a single partition, IQ-TREE spent most of the time applying ModelFinder. On the other hand, when we used IQ-TREE with 3 partitions, we had also used the option Merge, which tried to merge some partitions in the same evolutionary model.

ModelFinder is a tool that computationally depends on the alignment length and the number of partitions considered, but when IQ-TREE treats with mixtures, it parallelizes the ModelFinder work. Using this strategy, IQ-TREE with partitions can perform ModelFinder in less time than needed with only one. On the other hand, the option Merge also spends a lot of time, making IQ-TREE take longer when it considers the three partitions.

4.2 Study of SVD+WO

Once we obtained the previous good results on quartets for the implemented SVD algorithm, the further step was to study the performance of SVD with a quartet-based method (in our case: Weight Optimization) when applied to reconstruct trees for any number of leaves. Then, we adapted WO from an existing code ([Sab14]) to be able to incorporate the system of weights of SVD (see Algorithm 3.3).
This was also implemented in C++ and DendroPy library (Python, [SH10]) was used to compute the majority rule consensus tree.

To be able to evaluate the similarity of the inferred tree compared to the original one, we consider the following distance:

4.2.1 Definition (Robinson-Foulds distance) The *Robinson-Foulds distance* between two trees is the sum of the number of splits found in one of the trees but not in the other one.

Thus, this distance gives always an even number which is at most the double of the number of internal edges on a trivalent tree on n taxa. For example, between two trivalent trees with 19 leaves the distance is at most 32.

First, we will detail below how the data were simulated. Then, we will describe the different strategies performed to reconstruct the different alignments. Finally, the obtained results will be discussed.

To generate more realistic simulations, we simulated the alignments based on the structure of our real data. Then, we applied some modifications in order to hinder the reconstruction work, so the drawbacks of the used methods could be studied.

4.2.1 Simulated data

Angiosperms, commonly known as flowering plants, are morphologically classified in *monocotyledons* and *dicotyledons*. The former are characterized by seeds typically containing only one embryonic leaf (or *cotyledon*), while the latter has two cotyledons when starting growing. However, it is well known that dicotyledons do not reflect real evolutionary relationships since they never show reciprocal monophyly in phylogenetic studies, i.e., they do not share a common ancestor.

Persea americana is an avocado species that can be classified as a dicotyledon based on morphology. Previous phylogenetic studies (unpublished) using sequences from these species have suggested different positions of this species in the tree of Angiosperms, depending on the data and the reconstruction method used. In the light of these results, *Persea americana* has three possible phylogenetic locations, i) in the base of dicotyledons (tree $\mathcal{T}_1$ in Figure 4.3), ii) in the base of Angiosperms (tree $\mathcal{T}_2$ in Figure 4.3) or, iii) in the base of dicotyledons (tree $\mathcal{T}_3$ in Figure 4.3).

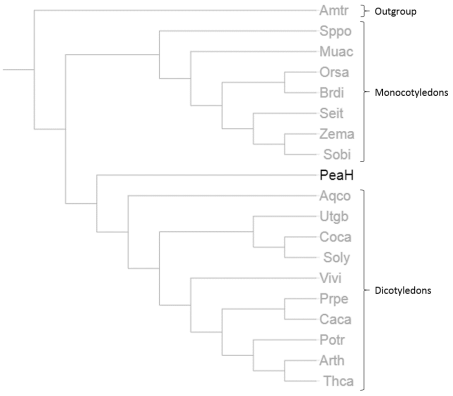
We simulated different alignments under these three possible topologies and those alignments were based on real data (provided by the Avocado Genome Consortium). More precisely, we have a collection of 189 alignments, each one corresponding to a protein alignment of the 19 considered species. We will refer to the different species by abbreviations (see Table 4.7), for example, PeaH is for *Persea americana*.

To know where to root the inferred trees, we will consider *Amborella trichopoda* (Amtr) as the outgroup.

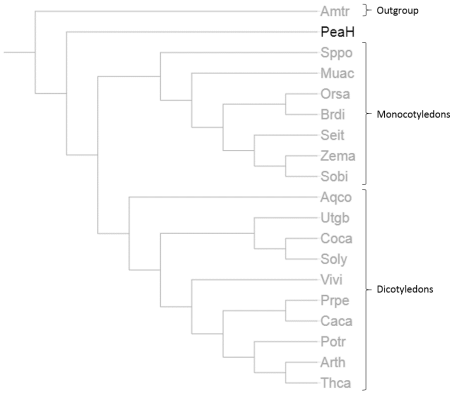
Species	Abbreviations	Phylogenetic Groups
Amborella trichopoda	Amtr	Outgroup
Aquilegia coerulea	Aqco	Dicotyledon
Arabidopsis thaliana	Arth	Dicotyledon
Theobroma cacao	Thca	Dicotyledon
Populus trichocarpa	Potr	Dicotyledon
Cajanus cajan	Caca	Dicotyledon
Prunus persica	Prpe	Dicotyledon
Vitis vinifera	Vivi	Dicotyledon
Coffea canephora	Coca	Dicotyledon
Solanum lycopersicum	Soly	Dicotyledon
Utricularia gibba	Utgb	Dicotyledon
Persea americana	PeaH	To determine
Brachypodium distachyon	Brdi	Monocotyledon
Oryza sativa	Orsa	Monocotyledon
Setaria italica	Seit	Monocotyledon
Sorghum bicolor	Sobi	Monocotyledon
Zea mays	Zema	Monocotyledon
Musa acuminata	Muac	Monocotyledon
Spirodela polyrrhiza	Sppo	Monocotyledon

Table 4.6: Abbreviations and phylogenetic groups of the 19 considered species. Notice that *Persea americana* is a dicotyledon based on its morphology, but its phylogenetic group is unknown.

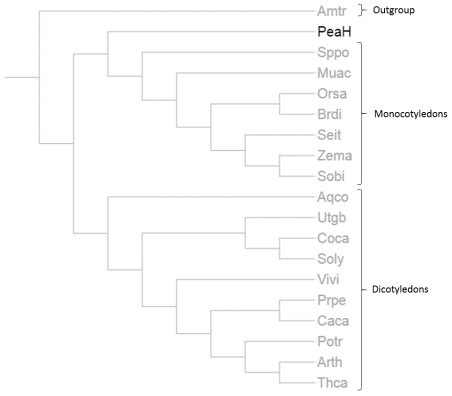
In the following Figure 4.3, the three possible considered topologies are shown:



(a) The topology $\mathcal{T}_1$ which considers PeaH as an outgroup of the dicotyledons.



(b) The topology $\mathcal{T}_2$ which considers PeaH as an outgroup of the monocotyledons and dicotyledons.



(c) The topology $\mathcal{T}_3$ which considers PeaH as an outgroup of the monocotyledons.

Figure 4.3: Three different possible topologies depending on the evolutionary location of *Persea americana*.

Two methodologies were followed in order to estimate the branch lengths and the rate matrices of the evolutionary model for each protein under each of the three topologies:

Rate matrices. Each of the 189 alignments was split into three sub-alignments in order to determine the rate matrices of the edges, assuming a heterogeneous model across lineages with at most three different rate matrices.
More precisely, the first sub-alignment was composed of the Amtr, PeaH and Aqco sequences[3]. The next consisted of the rest of the species belonging to monocotyledons and the last one consisted of the rest of species belonging to dicotyledons.

Therefore, for each of these three sub-alignments, we ran ModelFinder to determine the best evolutionary model relating the species of this sub-alignment. The branches that connect the different species of each sub-alignment were considered to have evolved under the estimated rate matrix of the Amtr, PeaH and Aqco sub-alignments.

A total of 12 different rate matrices was inferred from the total of 3*189 sub-alignments (see Table B.4 for further details). Moreover, JTT model was the chosen model in most of the sub-alignments.

Branch lengths. As IQ-TREE is designed to work only with unrooted trees (due to the non-identifiability of the root position, see Remark 3.0.2), we unrooted the three topologies accordingly. Thereby, all the simulated alignments were obtained following an unrooted tree, so, from now on, we will denote $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ as the three unrooted topologies. Moreover, for each one of the 189 alignments, we applied IQ-TREE to compute the maximum likelihood restricted to each of the three possible topologies (previously unrooted), i.e., we computed $\mathcal{L}(\hat{\theta})_{|\mathcal{T}_i}$ for every protein alignment, with $i \in \{1, 2, 3\}$. This approach allowed us to determine the most likely branch lengths under each topology.
Note that in this case, the maximum likelihood was calculated assuming the model was homogeneous across lineages, because IQ-TREE can not compute the likelihood for heterogeneous models. The data of the obtained branch lengths are not shown in this thesis due to its great dimensions.

In conclusion, for each protein and for each topology, we obtained an estimate of its branch lengths and its evolutionary model. So, we had all the parameters required to simulate new alignments under the three topologies. The next Figure 4.4 shows the workflow followed to obtain the branch lengths and the rate matrices of the evolutionary model of each protein. Recall that for topology $\mathcal{T}_3$ the branch length that connects the monocotyledons plus PeaH is almost null.

[3]It is necessary to have a minimum of three taxa in the alignments to apply ModelFinder. Although Aqco is a dicotyledon, we have considered this species as the third species because it is the more distant dicotyledon to the others.

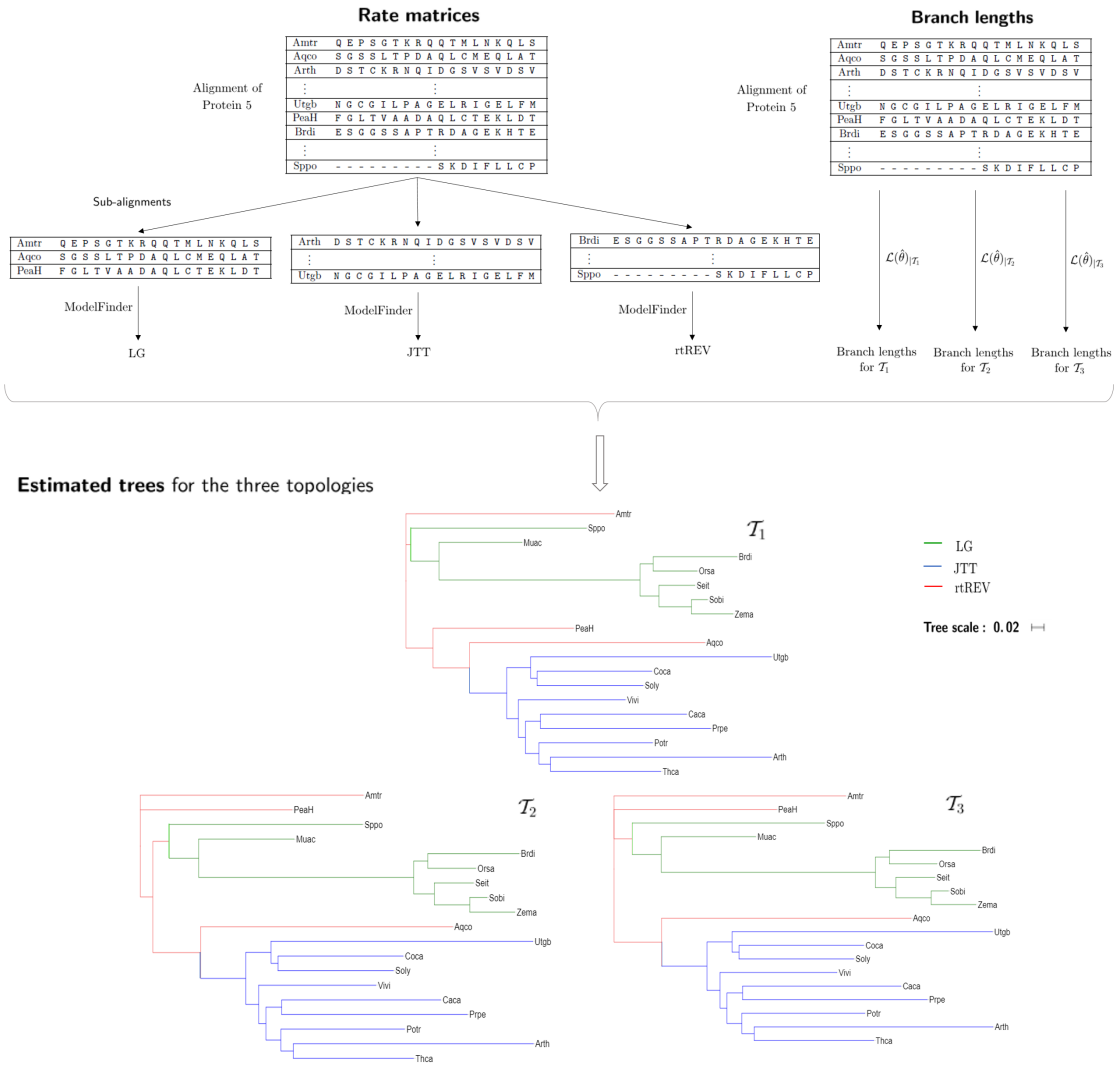


Figure 4.4: This workflow is an example of how the branch lengths and the rate matrices of the evolutionary model were inferred for each protein. In particular, it is shown for Protein 5.

We first simulated a set of alignments according to the parameters estimated from real data as above and then, in order to study more precisely the factors that affect the reconstruction performance for Algorithms 3.1 and 3.3, various parameters were modified in the simulations.

Alignments based on real data

For each of the three topologies, we generated 100 alignments which were composed of 189 partitions. The i-th partition was simulated under the evolutionary model with the three inferred rate matrices and the corresponding branch lengths. Moreover, the simulated i-th partition of the alignment had the same length as the original one, but

they did not have any gap (unlike real data). Therefore, each global alignment had length equal to the sum of the alignment lengths of the 189 proteins: 86244.

Modified alignments

Notice that assuming that the 189 proteins of real data had evolved according to similar evolutionary models and the same phylogenetic tree (the 189 proteins belong to the same partition) is quite unrealistic. If so, the same tree would be inferred independently of the reconstruction method or the data set, so the problem would be closed for its little difficulty. For this reason, due to the obtained results for alignments based on real data, we continued our study with new alignments with some modifications in the inferred parameters in order to study the performance of the SVD+WO with more difficult data sets. These new simulated alignments only covered the possibility of having different evolutionary models between the partitions.

Moreover, the various approaches considered allowed us to study in detail the behavior of the ML and SVD+WO reconstruction methods. The alignment lengths have been reduced so that the different reconstruction methods are faced with increasingly complicated data sets:

(A) **Alignments with half length.** We simulated three different sets of alignments with half the length of the original alignment:

A.1) First, we selected 10 protein alignments of the real data such that all the 12 different inferred rate matrices by ModelFinder were present. Each one had evolved under a heterogeneous across lineages model. Then, we generated 100 alignments for each of the three topologies following the same procedure as the alignments based on real data. In this case, the length of the simulated alignments of each partition was equal: overall, the concatenated alignments were $\frac{86244}{2} = 43122$ in length. Therefore, the first two simulated partition alignments had a length of $\lceil \frac{86244}{10 \cdot 2} \rceil = 4313$ and the rest of partitions had a length of $\lfloor \frac{86244}{10 \cdot 2} \rfloor = 4312$.

The motivation for these alignments is the possibility that the evolutionary processes of real data had evolved according to different rate matrices.

A.2) Now we considered a new set of alignments by permuting the columns of the alignments simulated in the previous point A.1. Therefore, we did not know which partition each site belonged to and applying Γ distribution for rates across sites models should become almost an useless strategy.

The motivation for these alignments is the possibility that close sites of the real data have not evolved according to the same evolutionary model.

A.3) Assuming time-reversibility as explained in Remark 2.2.3, a rate matrix is determined by a product of matrices $Q = RD(\pi^r)$. In this case we created an artificial rate matrix Q by taking R a matrix with entries in a sequential series (see figures A.6 and A.7). We call this new rate matrix Q_1.

Then we simulated 100 alignments for each of the three topologies with half length than the original one and assuming that they had evolved under a homogeneous model across lineages used by the rate matrix Q_1.

Protein	Q for edges joining monocotyledons	Q for edges joining dicotyledons	Q for the rest of the edges
5	LG	JTT	rtREV
15	LG	LG	rtREV
43	JTT	cpREV	FLU
105	WAG	JTT	cpREV
124	JTT	LG	VT
127	mtInv	mtMet	mtZOA
133	JTTDCMut	WAG	cpREV
137	cpREV	LG	WAG
151	WAG	JTTDCMut	PAMDCMut
179	JTT	LG	VT

Table 4.7: Inferred rate matrices from the 10 selected protein alignments assuming a heterogeneous model across lineages with at most three different rate matrices.

The motivation for these alignments is the possibility that the evolutionary processes of real data had evolved according to other rate matrices than considered in the previous studies.

(B) **Alignments with 10 times less length.** We simulated three different sets of alignments with 10 times less length than the original alignment:

B.1) This point followed the same procedure as the previous point A.1. In this case, the length of the simulated alignments of each partition was determined so that the concatenated alignment had length $\lfloor \frac{86244}{10 \cdot 10} \rfloor = 862$.

These alignments considered the same conditions as A.1, but their shorter length will determine a more complicated reconstruction work.

B.2) This point followed the same procedure as point A.2. In this case, the alignments were obtained by permuting the ones in point B.1.

These alignments considered the same conditions as A.2, but their shorter length will determine a more complicated reconstruction work.

B.3) As in point A.3, we created two additional new rate matrices, denoted by Q_2 and Q_3 (filling in R_2 and R_3 matrices entries permuting R_{JTT} and R_{cpREV}, respectively, see figures A.8 and A.9 for Q_2, and see figures A.10 and A.11 for Q_3). Then we simulated 100 alignments for each of the three topologies with 10 times less length than the original one. Every generated alignment had only one partition, which followed a heterogeneous model across lineages, where the rate matrices that described the evolutionary processes were Q_1, Q_2 or Q_3. More precisely, those matrices were alternated so that they described one of the three speciations mentioned: between monocotyledons, between dicotyledons and between the rest of the species. For example, for every topology, we obtained 34 alignments with Q_1 on the edges joining monocotyledons, Q_2 on the edges joining dicotyledons and Q_3 on the rest of the edges (for each of the other two cases there were 33 alignments).

The motivation for these alignments is the possibility that the evolutionary processes of the proteins of real data had evolved according to rate matrices that have not yet been considered and under heterogeneous evolutionary models.

(C) **Alignments with 10 times less length and modified branch lengths.** We simulated three different sets of alignments with 10 times less length than the original alignment and such that the partitions of a simulated alignment did not have the same branch lengths.
To this end, for every topology, we modified the branch lengths of the Protein 1. More precisely, the branch lengths of edges joining monocotyledons, of edges joining dicotyledons and the rest of the edges were multiplied each one by a scale factor (see Table 4.8).

	Branch lengths scale factor of		
	edges joining monocotyledons	edges joining dicotyledons	the rest of the edges
Partitions	1	1	1
	4	3	2
	2	3	4
	5	2	2
	2	5	5
	2	6	3
	4	1	3
	3	2	5
	5	1	1
	2	5	1

Table 4.8: Three scale factors for each (row) partition that were multiplied to the estimated branch lengths of the Protein 1.

C.1) This point followed the same procedure as the previous point B.1. In this case, the length of the simulated alignments of each partition was determined so that the concatenated alignment had a length of $\lfloor \frac{86244}{10 \cdot 10} \rfloor = 862$ and the simulated alignments of the i-th partition followed the evolutionary tree estimated for the Protein 1 with branch lengths multiplied by the scale factors specified in the i-th row of Table 4.8.

The motivation for these alignments is the possibility that the evolutionary processes of real data had evolved according to different rate matrices and not always under the same ones. Moreover, they also considered the possibility that the mutation rate between different phylogenetic groups and different proteins could differ.

C.2) This point followed the same procedure as point B.2. In this case, the alignments were obtained by permuting the ones in point C.1.

These alignments considered the same conditions as C.1 and also tried to consider the possibility that close sites of the real data had not evolved following the same evolutionary model.

C.3) (mixtures) We simulated 100 alignments for each topology with 10 times less length than the original alignment such that they followed a 10-mixture, i.e., the 10 simulated partition alignments were concatenated in order to obtain the whole one. We established that each partition had evolved under a heterogeneous

across lineages model where the rate matrices that described the evolutionary processes were Q_1, Q_2 or Q_3. More precisely, those matrices were alternated so that Q_1, Q_2 and Q_3 always described one of the three speciations mentioned in the previous point B.3.

4.2.2 Reconstruction strategies

We performed three main reconstruction strategies. Recall that in the first two following strategies WO was computed 1000 times and then returned the majority rule consensus tree. Moreover, each strategy was performed in different manners depending on the number of partitions considered:

(1) Using the SVD+WO algorithm implemented (Algorithm 3.3).

1.1. Considering only one partition, i.e., $r = 1$.

1.2. Considering four partitions, i.e., $r = 4$.

1.3. Considering twelve partitions, i.e., $r = 12$.

1.4. Considering the maximum possible of partitions, i.e., $r = 19$.

(2) Using the SVD+WO algorithm implemented (Algorithm 3.3) with a modified version of WO: it started with a random initial quartet q such that its weight was greater than 0.6, i.e., $w(q) > 0.6$.

2.1. Considering only one partition, i.e., $r = 1$.

2.2. Considering four partitions, i.e., $r = 4$.

2.3. Considering twelve partitions, i.e., $r = 12$.

2.4. Considering the maximum possible of partitions, i.e., $r = 19$.

The procedure of choosing a random initial quartet q with $w(q) > 0.6$ was designed in order to study how this initial quartet and the majority rule consensus tree affect the reconstruction process. The constriction $w(q) > \mathcal{E}$ was taken with $\mathcal{E} = 0.6$, because, on the one hand, a larger number could be a severe restriction so that WO could only start with a fewer variety of quartets and, on the other hand, a smaller number would not have the same confidence in choosing the right quartet.

(3) Using the ML algorithm (Algorithm 3.1) of the IQ-TREE software.

3.1. Considering only one partition. Then, ModelFinder was used in the same way as alignments with no mixtures.

3.2. Considering all possible partitions, i.e., 189 partitions for alignments based on real data and 10 partitions for points A.1, B.1, C.1 and C.3. In this case, ModelFinder+Merge could not be computed for computational time reasons (see Remark 3.1.1), and only ModelFinder was applied for each partition.

Recall that, unlike Algorithm 3.1, Algorithm 3.3 treats with r-mixtures only specifying the number of partitions.

4.2.3 Results and discussion

As above, we differentiate between simulated alignments based on real data and the set of alignments with parameters modified in their simulations. In the figures of this section, the results of each set of simulated alignments (described in section 4.2.1) are presented with a bubble chart, where the abscissa indicates the three different topologies and the ordinate the obtained Robinson-Fould distances. In addition, the colour indicates the number of partitions considered. Lastly, a greater thickness of the bubble describes a greater number of alignments with the same distance.

Alignments based on real data

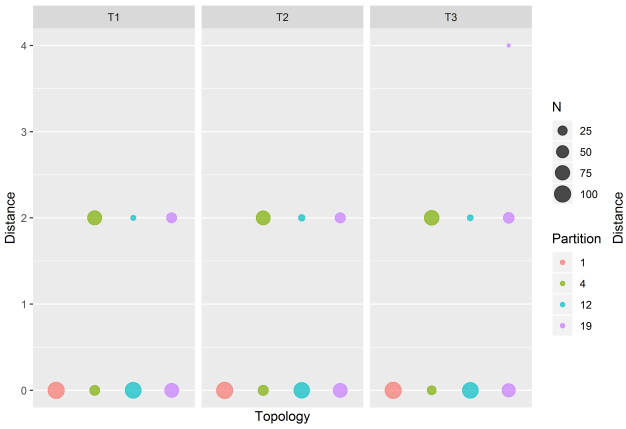
As Figure 4.5a and Figure 4.5b illustrate, the three topologies of simulated alignments based on real data were perfectly inferred by SVD+WO considering only one partition (i.e., $r = 1$), independently of the choice of the initial quartet that WO considers.

On the other hand, SVD+WO software could not reconstruct correctly all the alignments considering four, twelve or nineteen partitions. This suggests that although the alignments were performed concatenating 189 simulated partitions, the partitions could be considered as the same, i.e., the inferred branch lengths from real data and the rate matrices of each evolutionary process were not much different. Thereby, the rank of the flattening matrices may have not been affected.

It is worth pointing out that both figures show similar results for different topologies. Apparently, considering 12 partitions is a better approach than considering 19 partitions, which is also a better approach than considering 4 partitions. Moreover, the alignments that were not inferred correctly ($r = 4, 12, 19$) only differed in one split regarding to the correct topology, i.e., their Robinson-Foulds distances were 2.

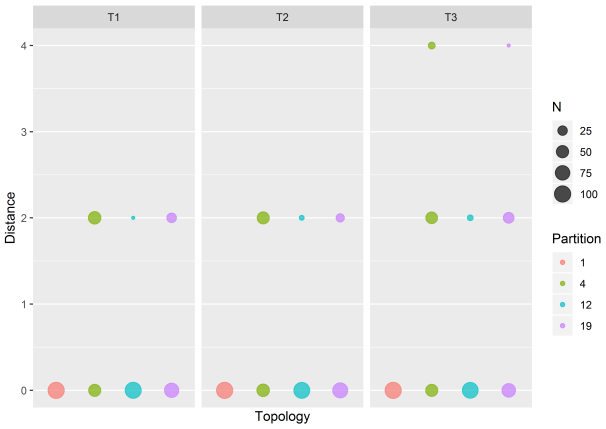
In addition, if we compare same topologies and same number of partitions (considering $r = 4, 12, 19$), SVD+WO software performs significantly better when choosing a random initial quartet q with $w(q) > 0.6$ (strategy 2) than the case when the initial quartet is chosen completely randomly (strategy 1). These obtained results make us wonder whether increasing the number of WO performances, the reconstruction strategy 1 results could be similar to the reconstruction strategy 2 results.

On the other hand, IQ-TREE performed correctly all the simulated alignments, regardless of whether it considered one or 189 partitions. This is due to the fact that if the evolutionary models for each partition are quite similar (furthermore, the vast majority of the considered rate matrices were JTT), ML could perfectly compute the correct topology assuming only one partition.



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
4	0	0	0	0	0	0	0	0	0	0	0	4
2	0	71	6	30	0	69	10	31	0	78	8	36
0	100	29	94	70	100	31	90	69	100	22	92	60
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(a) Reconstruction strategy 1.



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
4	0	0	0	0	0	0	0	0	0	9	0	1
2	0	52	1	26	0	48	3	17	0	43	5	35
0	100	48	99	74	100	52	97	83	100	48	95	64
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(b) Reconstruction strategy 2.

Figure 4.5: Robinson-Foulds distances for the simulated alignments based on real data according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software.

Modified alignments

(A) **Alignments with half length.** We present here the results on the three different sets of alignments with half the length of the original alignment:

A.1) In this case, as Figure 4.6a and Figure 4.6b show, the best results were obtained considering 12 partitions. So, the facts of halving the length of the alignments and not considering the exact numbers of partitions have determined that all the alignments were not correctly inferred.

Moreover, both figures show a slightly better performance for topology $\mathcal{T}_1$ in comparison with $\mathcal{T}_2$, and $\mathcal{T}_3$ was the worst one. Recall that $\mathcal{T}_1$ is the expected topology according to the morphology of PeaH.

As in the case of alignments based on real data, we see that the worst results were obtained considering $r = 4$ partitions, because they always failed inferring the correct topology (in this case, the Robinson-Foulds distance was 2 for all the alignments).

In both reconstruction strategies, we can see that $r = 19$ is the only case where some reconstructed trees gave a Robinson-Foulds distance greater than 2, though more than one in three times the tree was correctly inferred.

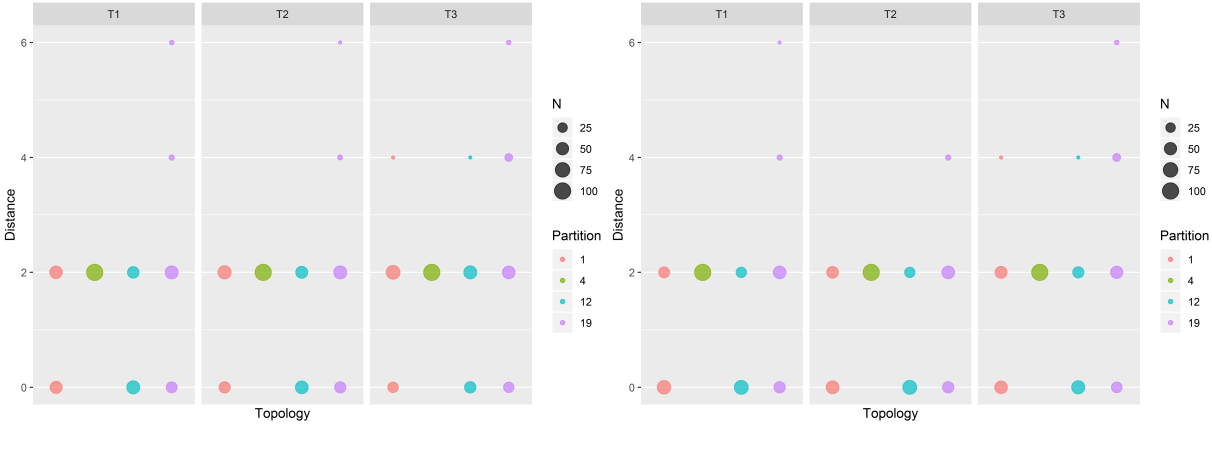
According to the reconstruction strategies, we see a significant better performance when WO chose the random initial quartet q such that $w(q) > 0.6$ (especially for $r = 1$ and $r = 12$ partitions).

On the other hand, IQ-TREE correctly performed all the simulated alignments, regardless of whether it had considered one or 10 partitions. In the next point

A.2, these results will be discussed together with the alignments corresponding to the point A.2 itself.

A.2) The same weighted quartets were obtained due to the fact that permutations of the alignments do not change the obtained flattening matrices. Then, almost the same results were obtained for SVD+WO software, regardless of the strategy to choose the initial quartet for WO. The few results that differed with A.1 alignments were due to the initial quartet, but the other replicas gave exactly the same result because the majority rule consensus tree was obtained with a high number (1000) of trees. Hence, results are not shown in new figures, as Figures 4.6a and 4.6b can lead to the same conclusions.

As far as IQ-TREE is concerned, its performance was perfect. It correctly reconstructed all the A.1 alignments (also when it considered 10 partitions) and the permuted ones. So, although it considered only one partition, it was possible to compute the optimal topology assuming the same rate matrix for all evolutionary processes. This fact supports that the rate matrices were quite similar and also the branch lengths of each partition were similar and near to zero, so each evolutionary process was always described by a transition matrix similar to the Id, making easier the process of reconstruction. For this reason, when IQ-TREE considers a single partition, ModelFinder determines the same rate matrix and the same rate classes for the discontinuous Γ distribution for each alignment and its permutation.



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
6	0	0	0	2	0	0	0	1	0	0	0	2
4	0	0	0	4	0	0	0	3	1	0	1	13
2	52	100	42	57	60	100	46	57	66	100	58	54
0	48	0	58	37	40	0	54	39	33	0	41	31
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(a) Reconstruction strategy 1.

	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
6	0	0	0	1	0	0	0	0	0	0	0	2
4	0	0	0	4	0	0	0	4	1	0	1	14
2	37	100	32	54	44	100	31	55	47	100	39	50
0	63	0	68	41	56	0	69	41	52	0	60	34
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(b) Reconstruction strategy 2.

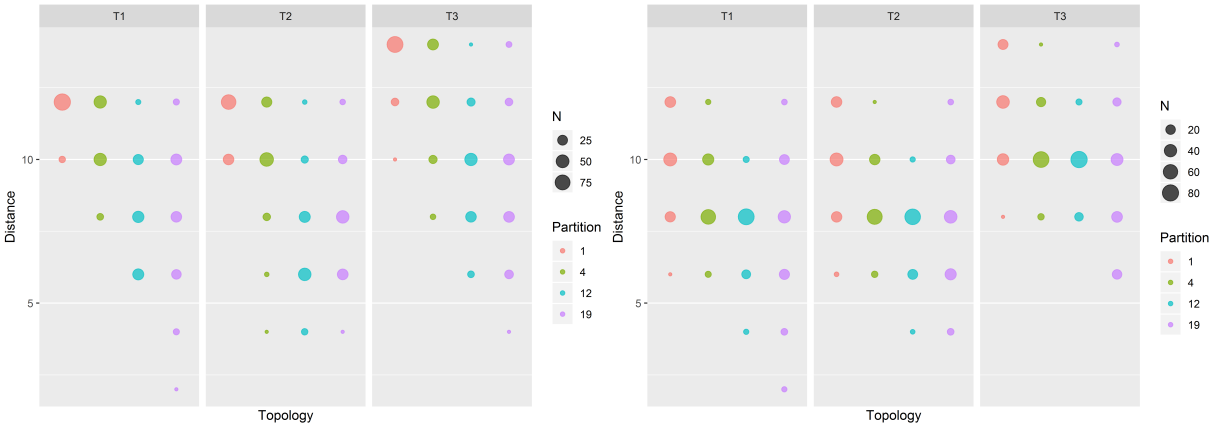
Figure 4.6: Robinson-Foulds distances for the simulated alignments of point A.1 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software.

A.3) So far, for simulated alignments such that all their evolutionary processes had evolved following the rate matrix Q_1, Figure 4.7a and Figure 4.7a present the worst results. These results took a large number of values because

the substitution probabilities that Q_1 determines make that any change between amino-acids were possible, so a greater number of different columns could be observed in the alignments, making that the rank of some flattening matrices was difficult to differentiate, or just the opposite, making it easier.

If both figures are compared, it emerges the importance of the initial quartet chosen by WO. In addition, the first tests of SVD+WO were carried out computing the majority rule consensus tree with only 100 replicas of WO, but the results obtained for these alignments were extremely poor, so we increased the times that WO was computed for all the simulated alignments.

On this occasion, we conclude that the topology $\mathcal{T}_2$ had the best results and $\mathcal{T}_3$ the worst and, despite following a single partition, the better performance was highlighted considering $r = 12$ partitions. Even so, when $r = 19$ partitions were considered, the range of the obtained distances was greater, appearing outliers that have values of 4 or 2 of distance.



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
14	0	0	0	0	0	0	0	0	87	33	1	5
12	93	46	3	6	70	27	2	4	12	48	14	12
10	7	46	27	33	30	59	10	16	1	15	46	33
8	0	8	36	31	0	11	34	47	0	4	31	32
6	0	0	34	23	0	2	47	32	0	0	8	17
4	0	0	0	6	0	1	7	1	0	0	0	1
2	0	0	0	1	0	0	0	0	0	0	0	0
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(a) Reconstruction strategy 1.

	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
14	0	0	0	0	0	0	0	0	23	1	0	2
12	28	3	0	4	27	1	0	4	43	18	5	13
10	46	31	5	22	45	26	3	15	33	75	81	37
8	25	61	76	40	26	67	73	43	1	6	14	28
6	1	5	16	24	2	6	22	31	0	0	0	20
4	0	0	3	7	0	0	2	7	0	0	0	0
2	0	0	0	3	0	0	0	0	0	0	0	0
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(b) Reconstruction strategy 2.

Figure 4.7: Robinson-Foulds distances for the simulated alignments of point A.3 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software.

In this case, IQ-TREE failed in 68% simulated alignments that had evolved under the topology $\mathcal{T}_3$ (for the rest of topologies its performance was perfect). More precisely, the reconstructed topology of all these trees was $\mathcal{T}_1$ or $\mathcal{T}_2$. Two possible circumstances could explain it:

(i) In fact, real data had not evolved following the topology $\mathcal{T}_3$, since it would be in the group of monocotyledons. For this reason, for topology $\mathcal{T}_3$ the branch length that connects the monocotyledons plus PeaH was almost null. Then, when IQ-TREE tried to reconstruct the alignments of the topology

$\mathcal{T}_3$, a phenomenon of long branch attraction was produced (and the same with SVD+WO).

(ii) On the other hand, we could be observing what happens in the real case: IQ-TREE (and also SVD+WO) was unable to reconstruct properly the phylogenetic location of PeaH because the evolutionary model was incorrectly inferred by ModelFinder. Indeed, the rate matrix inferred for all the edges was the so-called PMB (Probability Matrix from Blocks, [VST03]), a revised BLOSUM matrix. Recall that BLOSUM (and also PMB) is not recommended as it was designed mainly for sequence alignments.

(B) **Alignments with 10 times less length.** We present here the results on the three different sets of alignments with 10 times less length than the original alignment:

B.1) Although the alignments of this point were shorter than the ones of the point A.1, they were simulated following the same procedure, so these two points share analogies. Even so, Figure 4.8a and Figure 4.8b illustrate that having a shorter alignment determines a worse performance.

In particular, both A.1 and B.1 had the best performance considering $r = 12$ partitions. Recall that the SVD+WO was not computed considering $r = 10$ partitions, which was the number of expected partitions. Moreover, both figures show a slightly better performance for topology $\mathcal{T}_1$ in comparison with $\mathcal{T}_2$, and $\mathcal{T}_3$ was the worst.

What is most remarkable about this point is that our software could not compute the quartet weights when it considered $r = 19$ partitions. This was because the rank of none of the three flattening matrices for the three possible topologies had the maximum rank as expected. More precisely, for every 4-tuple, the three Frobenius-distances between the flattening matrices and the matrix space $E_{19\cdot 20}$ were equal to 0, i.e., $d_F(Flatt_{12|34}(\rho), E_{19\cdot 20}) = d_F(Flatt_{13|24}(\rho), E_{19\cdot 20}) = d_F(Flatt_{23|14}(\rho), E_{19\cdot 20}) = 0$ (see equation 3.1). Thereby, the weight of these three quartets could not be calculated as the denominators of their expressions were 0, so WO could not be performed.

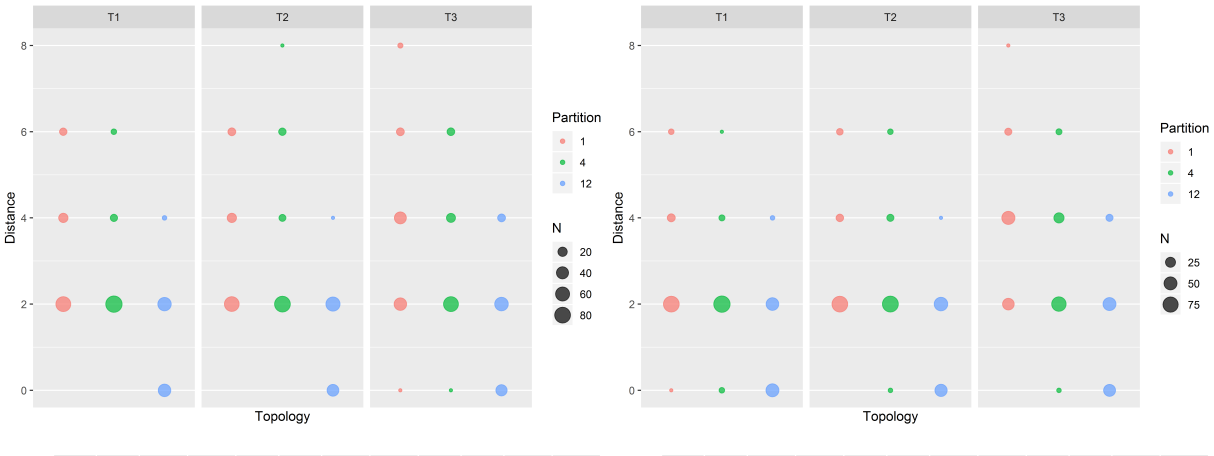
According to the reconstruction strategies, as previous results, we see a significant difference when WO chooses the random initial quartet q such that $w(q) > 0.6$.

On the other hand, IQ-TREE correctly performed on all the simulated alignments, regardless of whether it considered a single partition or $r = 10$ partitions.

B.2) As has been mentioned in A.1, permutations of the columns do not change the weights of the quartets. Then, just a few results differed with B.1 alignments, due to the initial quartet, and the results are not shown in new figures.

As far as IQ-TREE is concerned, its performance was perfect (also for alignments of point B.1), for the same reasons explained in the point A.2.

B.3) As in the previous case of alignments that had evolved under an artificial rate matrix, Figure 4.9a and Figure 4.9b presented again the worst results so far. Also, the alignment length played the decisive role to give these great Robinson-Foulds distances. On the other hand, as we saw in section 4.1.3 (results of heterogeneous models), the fact that the alignments had evolved under a heterogeneous across lineages model had a low impact.



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
8	0	0	0	0	0	1	0	0	3	0	0	0
6	10	4	0	0	12	10	0	0	12	11	0	0
4	20	9	2	0	20	8	1	0	39	18	12	0
2	70	87	53	0	68	81	61	0	45	70	55	0
0	0	0	45	0	0	0	38	0	1	1	33	0
TOTAL	100	100	100	0	100	100	100	0	100	100	100	0

(a) Reconstruction strategy 1.

	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
8	0	0	0	0	0	0	0	0	1	0	0	0
6	4	1	0	0	7	4	0	0	9	5	0	0
4	13	5	2	0	11	9	1	0	52	26	9	0
2	82	90	47	0	82	85	56	0	38	67	50	0
0	1	4	51	0	0	2	43	0	0	2	41	0
TOTAL	100	100	100	0	100	100	100	0	100	100	100	0

(b) Reconstruction strategy 2.

Figure 4.8: Robinson-Foulds distances for the simulated alignments of point B.1 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software.

Lastly, same conclusions as alignments of A.3 were obtained: the range of the obtained distances was greater when $r = 19$ partitions were considered, the results improved when WO took an initial quartet q with $w(q) > 0.6$, the best results were fulfilled for $\mathcal{T}_2$ and when 12 partitions were considered (though the alignments consist of a single partition).

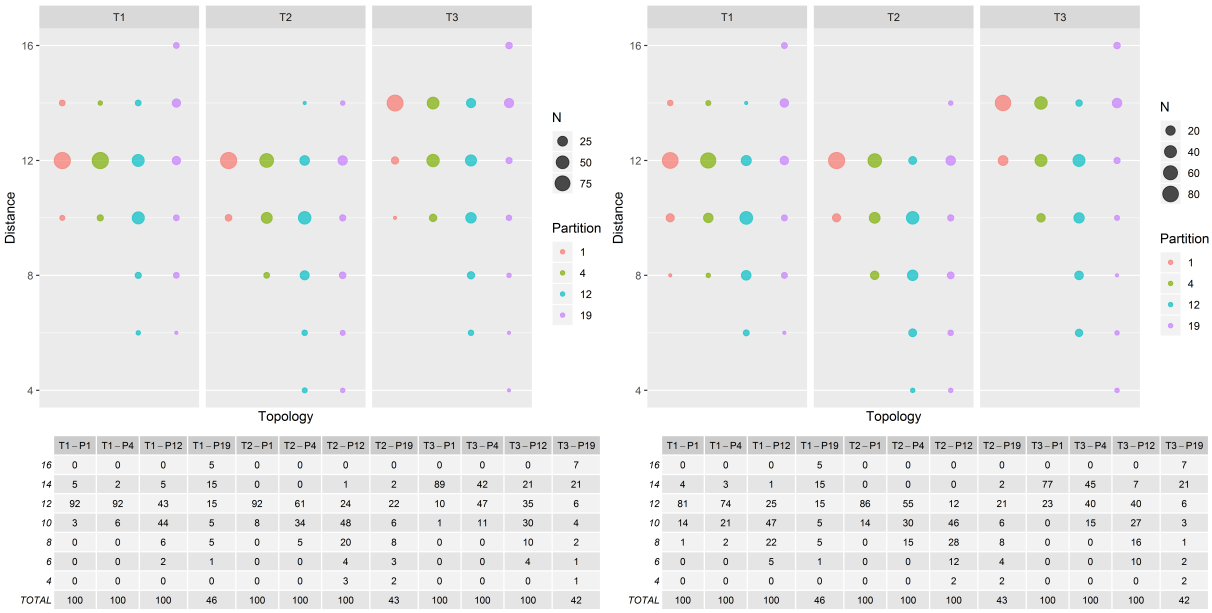
Moreover, for SVD+WO, topology $\mathcal{T}_3$ had the worst results (probably because we had a case of long branch attraction).

On this occasion, IQ-TREE failed all the simulated alignments that had evolved under the topology $\mathcal{T}_3$ (for the rest of topologies its performance was perfect). The topology reconstructed of all these trees was $\mathcal{T}_1$ or $\mathcal{T}_2$, so it failed the phylogenetic location of PeaH. In addition, as A.3, the inferred rate matrix for all the edges was PMB.

(C) **Alignments with 10 times less length and modified branch lengths.** We present here the results on the three different sets of alignments with 10 times less length than the original alignment and with the modified branch lengths:

C.1) In Figure 4.10 we show the results for the simulated data C.1. Despite not considering the exact number of topologies for SVD+WO (10-mixture), if we compare Figures 4.8a and 4.8b with Figures 4.10a and 4.10b, only one main conclusion remains: when SVD+WO considered alignments such that their partitions had branch lengths highly different, its performance seriously worsened.

Moreover, Figures 4.10a and 4.10b show for first time clear better results for $\mathcal{T}_2$. Additionally, considering 19 partitions could be the best approach, if it were not



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
16	0	0	0	5	0	0	0	0	0	0	0	7
14	5	2	5	15	0	0	1	2	89	42	21	21
12	92	92	43	15	92	61	24	22	10	47	35	6
10	3	6	44	5	8	34	48	6	1	11	30	4
8	0	0	6	5	0	5	20	8	0	0	10	2
6	0	0	2	1	0	0	4	3	0	0	4	1
4	0	0	0	0	0	0	3	2	0	0	0	1
TOTAL	100	100	100	46	100	100	100	43	100	100	100	42

(a) Reconstruction strategy 1.

	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
16	0	0	0	5	0	0	0	0	0	0	0	7
14	4	3	1	15	0	0	0	2	77	45	7	21
12	81	74	25	15	86	55	12	21	23	40	40	6
10	14	21	47	5	14	30	46	6	0	15	27	3
8	1	2	22	5	0	15	28	8	0	0	16	1
6	0	0	5	1	0	0	12	4	0	0	10	2
4	0	0	0	0	0	0	2	2	0	0	0	2
TOTAL	100	100	100	46	100	100	100	43	100	100	100	42

(b) Reconstruction strategy 2.

Figure 4.9: Robinson-Foulds distances for the simulated alignments of point B.3 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software.

for its unstable results.

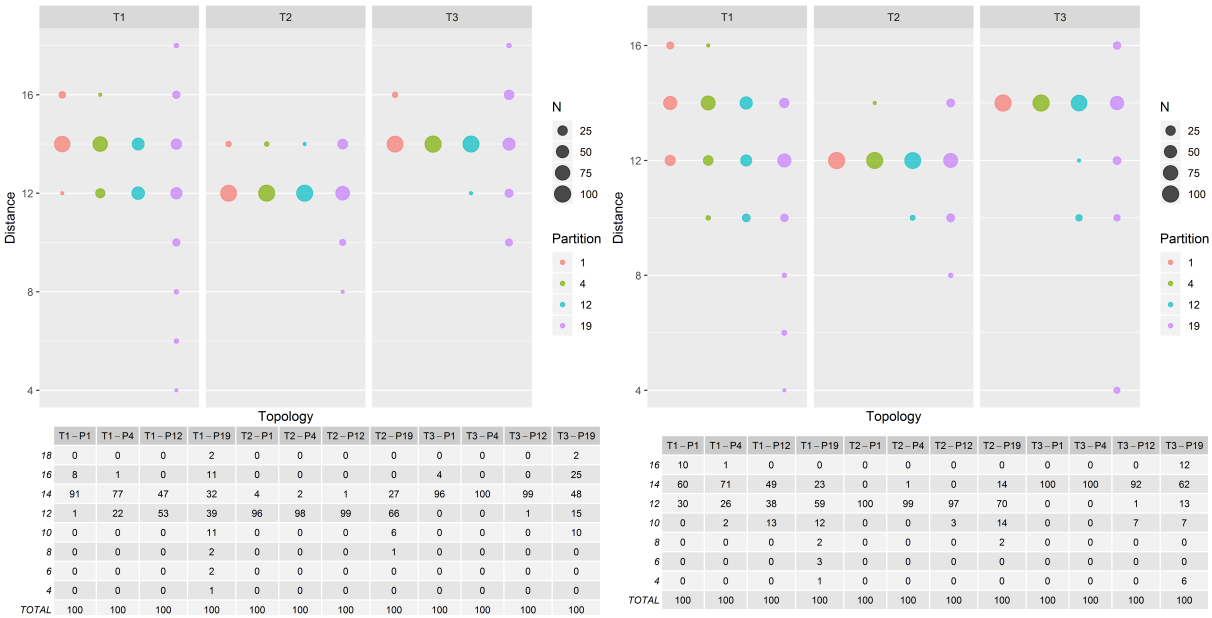
The rest of findings were exactly the same ones mentioned in points B.1 and B.2.

C.2) As has been mentioned, permutations of the columns almost did not change the results. Then, they are not shown in new figures, as Figures 4.10a and 4.10b can lead to the same conclusions.

As far as IQ-TREE is concerned, its performance failed in 99% simulated alignments that had evolved under the topology $\mathcal{T}_3$ (for the rest of topologies its performance was perfect). More precisely, the exact same results were obtained regardless of the considered number of partitions and regardless of whether the alignments were permuted: only one case was reconstructed correctly and another was reconstructed with a Robinson-Foulds distance of 4. All remaining alignments gave a distance of 2 due to the phylogenetic position of PeaH.

Hence, so far we have seen that IQ-TREE only failed for topology $\mathcal{T}_3$, when the alignment length was reduced and either the evolutionary model was not among those considered by IQ-TREE, or the branch lengths of its partitions were quite different.

C.3) The results for the simulated data C.3 are shown in Figure 4.11. In essence, almost the same findings can be obtained by analyzing Figure 4.10a and Figure 4.11a. But, when these two figures and Figure 4.9a are compared, we can see the fact that the alignments had evolved by evolutionary models with rate matrices Q_1, Q_2, Q_3 and that their partitions had different branch lengths barely increased the Robinson-Foulds distances. In fact, the main difference is



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
18	0	0	0	2	0	0	0	0	0	0	0	2
16	8	1	0	11	0	0	0	0	4	0	0	25
14	91	77	47	32	4	2	1	27	96	100	99	48
12	1	22	53	39	96	98	99	66	0	0	1	15
10	0	0	0	11	0	0	0	6	0	0	0	10
8	0	0	0	2	0	0	0	1	0	0	0	0
6	0	0	0	2	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(a) Reconstruction strategy 1.

	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
16	10	1	0	0	0	0	0	0	0	0	0	12
14	60	71	49	23	0	1	0	14	100	100	92	62
12	30	26	38	59	100	99	97	70	0	0	1	13
10	0	2	13	12	0	0	3	14	0	0	7	7
8	0	0	0	2	0	0	0	2	0	0	0	0
6	0	0	0	3	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	6
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(b) Reconstruction strategy 2.

Figure 4.10: Robinson-Foulds distances for the simulated alignments of points C.1 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software.

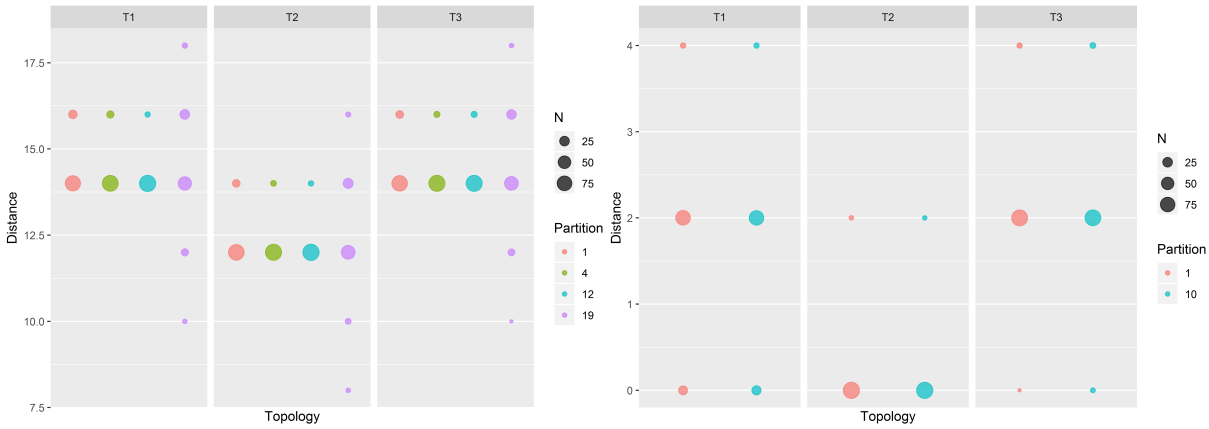
that results from alignments with these two modified parameters (from Figure 4.10a) were more homogeneous.

In respect of reconstruction strategy 2, WO could not be performed for any alignment because none of the quartet weights had a value greater than 0.6 ($w(q)$>0.6), so WO could not take any initial quartet q. This tells us that on this occasion the inferred topology for each 4-tuple did not have a much greater weight than the other two possible topologies, i.e., the assurance that the correct topology for each 4-tuple was correctly inferred is lower.

Regarding to IQ-TREE, unlike the previous cases, this software did not achieve to reconstruct perfectly the replicas of the three topologies (see Figure 4.11b), failing almost always in the phylogenetic location of PeaH. However, for the topology $\mathcal{T}_2$, IQ-TREE only failed in two cases. Furthermore, as previous results, there was a phenomenon of long branch attraction for almost all alignments in the case of $\mathcal{T}_3$.

Lastly, it is important to note that the results considering a single partition and the results considering $r = 10$ partitions barely differed. This is because ModelFinder always established PMB as the rate matrix of the evolutionary model, so only the mutation rate could differ (it depends on the strategy that IQ-TREE applies to model heterogeneity across sites), which was almost completely solved due to IQ-TREE applied a free rate model[4] when it considered a single partition. In other words, the mutation rate could be modelled similarly due to IQ-TREE

[4]A free rate model generalizes the +Γ model by relaxing the assumption of Γ-distributed rates



	T1−P1	T1−P4	T1−P12	T1−P19	T2−P1	T2−P4	T2−P12	T2−P19	T3−P1	T3−P4	T3−P12	T3−P19
18	0	0	0	4	0	0	0	0	0	0	0	2
16	17	10	4	25	0	0	0	3	13	6	6	24
14	83	90	96	59	12	5	4	27	87	94	94	64
12	0	0	0	10	88	95	96	63	0	0	0	9
10	0	0	0	2	0	0	0	5	0	0	0	1
8	0	0	0	0	0	0	0	2	0	0	0	0
TOTAL	100	100	100	100	100	100	100	100	100	100	100	100

(a) Reconstruction strategy 1.

	T1−P1	T1−P10	T2−P1	T2−P10	T3−P1	T3−P10
4	4	4	0	0	4	5
2	76	74	2	2	95	92
0	20	22	98	98	1	3
TOTAL	100	100	100	100	100	100

(b) Reconstruction strategy 3.

Figure 4.11: Robinson-Foulds distances for the simulated alignments of point C.3 according to their corresponding correct topologies. Reconstruction strategies 1 and 3 (reconstruction strategy 2 could not be performed).

applied five rate classes of the free discrete distribution when it considered a single partition and, for each partition in the case where $r = 10$ partitions were considered, it only applied four rate classes of a Γ discrete distribution.

Running time

When it comes to the computational time issue (under the same conditions: 2 CPUs and the same RAM available), SVD+WO is usually ahead of IQ-TREE. The average results of running time for one alignment (computed from a sample of 300, since there are 100 alignments for each possible topology) are shown in two different tables: Table 4.9 for alignments based on real data and Table 4.10 for modified alignments.

Firstly, in the case where IQ-TREE considers more than one partition, we tried to compute the best partition scheme for each alignment using ModelFinder+Merge. However, we stopped the computations before they finished, because the time that had elapsed was greater than 2 months for a single alignment based on real data and greater than 2 weeks for a single alignment with half length. Hence, all results shown for IQ-TREE are computed only with the option ModelFinder.

Focusing on SVD+WO software (reconstruction strategies 1 and 2), the computational time for SVD+WO software does not depend on the number of the considered partitions, because it only differs in the singular values taken when it calculates the weights. The running time for SVD+WO also does not depend on whether the alignments are permuted (as the joint distribution vector is the same) and on the considered rate

matrices. However, it does depend on the length of the alignment. Moreover, there is no difference between the treatment of the initial quartet q, because when SVD+WO only considers an initial quartet such that $w(q) > 0.6$, it usually finds it at the third or fourth quartet that is screened. As already discussed, recall that for alignments of point C.3, there were no quartets that fulfill this condition.
Furthermore, when alignments have evolved under partitions with different branch lengths, the computational time is longer. These results can be as follows. The substitution probabilities between two different amino-acids were much greater (since the branch lengths had a much greater value than the previous cases), so this makes that any set of amino-acids could appear in the alignments (even with amino-acids of different groups), and a greater number of different columns were observed in the alignments, making that more entries of the flattening matrices were filled, but with smaller values. For this reason, it took longer to build the joint distribution vector and the flattening matrices.

Although SVD+WO scales better for large inputs, when the alignments are short enough and the number of partitions considered is small, IQ-TREE runs in less time (see time of B.1, Table 4.10), because it is able to parallelize the work, especially the one of ModelFinder, which is what consumes more. For this reason, IQ-TREE reconstructs the phylogenetic tree in less time when it considers 189 partitions (or 10 for modified alignments). Moreover, if we had worked with more CPUs, IQ-TREE would have parallelized even more the reconstruction process and would have finished in less time than SVD+WO, due to SVD+WO software has not been designed to parallelize the work.

		Reconstruction strategies		
		SVD+WO strategy 1	SVD+WO strategy 2	IQ-TREE strategy 3
Alignments based on real data	Treating like one partition	5h 1’	5h 3’	19h 33’
	Treating like r partitions	5h 5’	4h 57’	8h 40’

Table 4.9: Average of computational time on one alignment for the three main reconstruction strategies (same hardware conditions). Results are shown depending on whether a single partition or more has been considered (in case of the SVD+WO, the results correspond to the average of one alignment treating it like $r = 4, 12$ and 19, and $r = 189$ in case of the IQ-TREE).

Note that the running time for IQ-TREE does not depend on whether the columns are permuted, because it determines the same rate matrix and the same rate classes for the discontinuous Γ distribution. On the other hand, the running time does increase when the alignment has evolved under artificial rate matrices. The reason is that it takes longer to converge as its initial phylogenetic tree (obtained by a distance method) is not so good as previous cases (due to the poor choice of the evolutionary model).

Lastly, it is important to notice that IQ-TREE spends much more time with alignments with modified branch lengths, because considering the free rate model that generalizes the $+\Gamma$ model with five rate classes is computationally expensive.

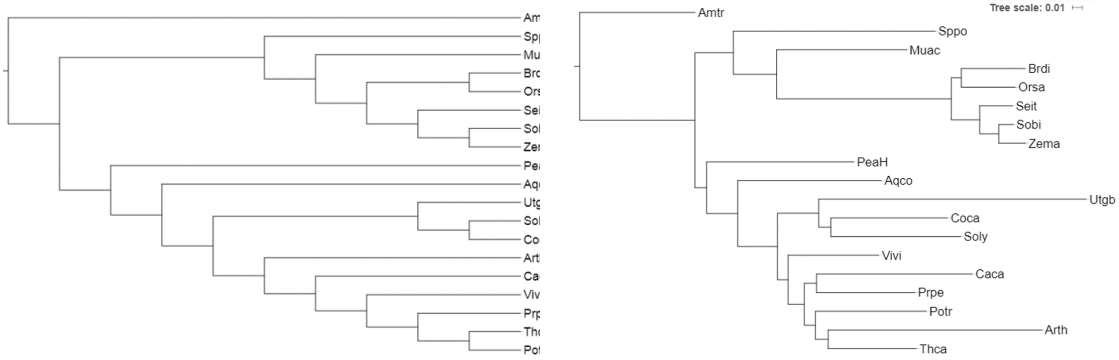
			Reconstruction strategies		
		Alignments of points	SVD+WO strategy 1	SVD+WO strategy 2	IQ-TREE strategy 3
Alignments with half length	Treating like one partition	A.1	3h 54'	3h 53'	9h 7'
		A.2	3h 58'	3h 54'	9h 4'
		A.3	3h 54'	3h 57'	10h 14'
	Treating like r partitions	A.1	3h 55'	4h	8h 40'
		A.2	3h 51'	3h 56'	-
		A.3	3h 50'	3h 54'	-
Alignments with 10 times less length	Treating like one partition	B.1	2h 54'	2h 54'	3h 33'
		B.2	2h 48'	2h 59'	3h 36'
		B.3	2h 55'	2h 51'	3h 55'
	Treating like r partitions	B.1	2h 54'	2h 53'	2h 1'
		B.2	2h 56'	2h 52'	-
		B.3	2h 54'	2h 52'	-
Alignments with 10 times less length and modified branch lengths	Treating like one partition	C.1	3h 15'	3h 18'	8h 27'
		C.2	3h 13'	3h 16'	8h 31'
		C.3	3h 19'	3h 15'	9h 19'
	Treating like r partitions	C.1	3h 13'	3h 12'	4h 12'
		C.2	3h 18'	3h 9'	-
		C.3	3h 20'	-	4h 40'

Table 4.10: Average of computational time on one alignment for SVD and IQ-TREE (same hardware conditions). Results are shown depending on alignment length and the partitions considered (in case of the SVD+WO, the results correspond to the average of one alignment treating it like $r = 4, 12$ and 19, and $r = 189$ in case of the IQ-TREE).

Proposed topology for real data

When real data are reconstructed using the two main reconstruction strategies of SVD+WO with the same number of partitions, the same results are obtained, i.e., the topology obtained does not depend on the choice of the initial quartet of WO. More precisely, when it considers a single partition or 4 partitions, SVD+WO places PeaH as an outgroup of the monocotyledons and dicotyledons (as $\mathcal{T}_2$) and, when it considers 12 partitions, it places PeaH in the base of dicotyledons (as $\mathcal{T}_1$, see Figure 4.12a). Furthermore, for different number of partitions ($r = 1, 4, 12$), the Robinson-Foulds distance between the topology reconstructed and the proposed one ($\mathcal{T}_2$ for $r = 1, 4$ and $\mathcal{T}_1$ for $r = 12$) is 8, because some species of dicotyledons are not reconstructed entirely well. In case of 19 partitions, the Robinson-Foulds distance to the three topologies is much greater (16) and it is impossible to conclude which of the three phylogenetic location describes.

On the other hand, IQ-TREE reconstructs exactly the topology $\mathcal{T}_1$ whether it considers one or 189 partitions. In addition, both reconstructed trees have similar branch lengths, so only that with a single partition is shown in the Figure 4.12b. Then, along with the fact that SVD+WO also places PeaH in the base of dicotyledons when it considers 12 partitions (which, as has been discussed above, is the one that gives the best results for SVD+WO), we proposed the topology $\mathcal{T}_1$, so PeaH belongs to the dicotyledons in terms of evolutionary relationships.



(a) Reconstructed topology with SVD+WO ($r = 12$).

(b) Reconstructed tree with IQ-TREE ($r = 1$).

Figure 4.12: Reconstructed trees using the three main reconstruction strategies, and considering $r = 12$ partitions for SVD+WO and a single partition for IQ-TREE. Note that PeaH is placed in the base of dicotyledons.

Conclusions

Throughout this project, we have adapted the algebraic phylogenetic reconstruction techniques based on SVD to amino-acid sequences. To this end, we have implemented the SVD software for a general alphabet based on software developed by [CFS16]. Accordingly, we have been able to generate protein alignments in order to test our software. Despite treating with sparse flattening matrices (due to their big size), our study leads to ensure that theoretical results based on algebraic tools can indeed be applied for phylogenetic reconstruction using amino-acid sequences.

In the course of the reconstruction study for an arbitrary number of species, we have adapted Weight Optimization (WO) from an existing code [Sab14]. In particular, we have customized the systems of weights to apply the ones obtained by SVD (SVD+WO) and we have designed a different strategy to choose the initial quartet. Moreover, we have studied its performance in phylogenetic reconstruction over different sets of alignments based on real-life data. These last results have allowed us to propose a phylogenetic tree for our real data.

In order to have a benchmark with a classical method, we have also applied a ML software (IQ-TREE) to all our simulated data. Moreover, as far as we are aware, we have been the first to perform a phylogenetic reconstruction method based on invariants for amino-acid sequences.

All of this has enabled us to reach the following conclusions:

- In the case of the quartet reconstruction, the reconstructed topologies of SVD had almost no failures (only a few cases with homogeneous models), and IQ-TREE performance was perfect. Nevertheless, the running time of IQ-TREE was considerably higher.

- Based on the results we have obtained, SVD+WO performance has a clear dependence with the treatment on the initial quartet and with the number of trees that the majority rule consensus tree considers. Furthermore, considering a number of partitions neither too big nor too small seems a good approach in the reconstruction performance.

- Alignments that have evolved under uneven substitution probabilities or under partitions with considerable different branch lengths are difficult to overcome.

- IQ-TREE software gives more accurate phylogenetic reconstructions than SVD+ WO software. On the other hand, the long branch attraction situations are hard to beat for both software.
- Despite its limitations, SVD+WO provides a good approach to the topology tree inference problem, providing fast and accurate tree inference. Additionally, unlike most statistical phylogenetic programs, the evolutionary model does not need to be specified before the analysis starts, Moreover, it also has the benefits that some aspects of the evolutionary processes can be much more general:

  (i) Our software can be applied to deal with complex evolutionary models such as phylogenetic mixtures (only specifying the number of partitions, without the need to know the specific position of the partitions).

  (ii) Our software accounts for non-homogeneous evolutionary processes for every partition of the sequence.

- As far as our real data is concerned, we have proposed the base of dicotyledons as the phylogenetic location of *Persea americana*.

Moreover, our project has led us to a wide range of unresolved subjects. The most relevant ones are comparing our method with a fast reconstruction phylogenetic method (like Neighbor-Joining, [SN87]), study the pattern of the flattening matrices obtained with protein alignments in order to apply a correction which can handle long branch attraction situations (like Erik+2, [CFS16]), and study the treatment of the initial quartet by WO and its quartet weighting (as [RG01a] demonstrated that it is the weakness of quartet-based methods).

List of Figures

1.1 The join of two nucleotides is based on the link between a phosphate group to their deoxyribose units. Taken from [HA05]. 4
1.2 Schematic diagram of the DNA double helical structure. Taken from [HA05]. 4
1.3 Chemical structure of an amino acid (a) and a protein backbone (b) composed of three amino-acids linked by peptide bonds (boxed). Taken from [HA05]. 5
1.4 The tree of life. The three major branches represent the three domains proposed by Woese (1990): bacteria, archaea and eukaryotes. Taken from [Pev15]. 11
1.5 The three possible topologies of an unrooted tree $\mathcal{T}$ of four leaves. . 12

2.1 A rooted tree $\mathcal{T}$ of four leaves $X_1, ..., X_4$. 19
2.2 A heterogeneity across lineages model on a rooted tree $\mathcal{T}$ of three leaves. Transition matrices for each evolutionary process are indicated along the branches. 19
2.3 Simplified phylogenetic tree inferred by maximum parsimony from an alignment of four amino-acid sequences. Inferred sequences are shown at the interior nodes. Amino-acid exchanges are indicated along the branches. In the sequence at the root, we indicate by brackets those positions where different amino-acids can be chosen. . 21
2.4 An alignment of 7 species (on the left) to obtain a symmetric matrix A (on the right). Its entries are the number of times the pairs of aligned amino-acids appeared in the data. 22

3.1 An evolutionary model on a rooted tree $\mathcal{T}$ of three leaves. Transition matrices for each evolutionary process are indicated along the branches. 26
3.2 The internal node u is the median of x, y, z. It splits the tree into three subtrees $\mathcal{T}_x$, $\mathcal{T}_y$ and $\mathcal{T}_z$. Every edge of $\mathcal{T}$ belongs to a single one of these subtrees. For example, if $e \in \mathcal{T}_z$, then its score is increased by the weight $w(xy \mid zi)$. 36
3.3 A reconstruction of a tree of five leaves using Weight Optimization algorithm ([RG01a]). 37

4.1 Reconstruction methods fail in the phenomenon of long branch attraction, where separated long branches are inferred as adjacent. . . 40

4.2 The 9 different unrooted trees that were considered in the simulations. 41
4.3 Three different possible topologies depending on the evolutionary location of *Persea americana*. 49
4.4 This workflow is an example of how the branch lengths and the rate matrices of the evolutionary model were inferred for each protein. In particular, it is shown for Protein 5. 51
4.5 Robinson-Foulds distances for the simulated alignments based on real data according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software. 57
4.6 Robinson-Foulds distances for the simulated alignments of point A.1 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software. 58
4.7 Robinson-Foulds distances for the simulated alignments of point A.3 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software. 59
4.8 Robinson-Foulds distances for the simulated alignments of point B.1 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software. 61
4.9 Robinson-Foulds distances for the simulated alignments of point B.3 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software. 62
4.10 Robinson-Foulds distances for the simulated alignments of points C.1 according to their corresponding correct topologies. Reconstruction strategies using SVD+WO software. 63
4.11 Robinson-Foulds distances for the simulated alignments of point C.3 according to their corresponding correct topologies. Reconstruction strategies 1 and 3 (reconstruction strategy 2 could not be performed). 64
4.12 Reconstructed trees using the three main reconstruction strategies, and considering $r = 12$ partitions for SVD+WO and a single partition for IQ-TREE. Note that PeaH is placed in the base of dicotyledons. 67

A.1 Chemical structure of the 20 amino-acids. They are classified by the properties of their residue group R into four groups: acidic, basic, nonpolar and polar amino-acids. Taken from [HA05]. 83
A.2 Number of accepted point mutations (multiplied by 10) accumulated from closely related sequences (symmetric matrix). Taken from [DSO78]. 84
A.3 Relative frequencies of the amino-acids in the Accepted Point Mutation Data. Taken from [DSO78]. 84
A.4 Transition matrix for the evolutionary distance of 1 PAM. To simplify its appearance, the entries are shown multiplied by 10 000. Taken from [DSO78]. 85
A.5 Transition matrix for the evolutionary distance of 250 PAMs. To simplify its appearance, the entries are shown multiplied by 100. Taken from [DSO78]. 86
A.6 Created R_1 matrix (symmetric matrix). The dots can be determined by the condition that all the Q_1 rows add up to 0. 87
A.7 Created π_1. It determines the matrix $D_1(\pi)$. 87

A.8 Created R_2 matrix (symmetric matrix). The dots can be determined by the condition that all the Q_2 rows add up to 0. 88
A.9 Created π_2. It determines the matrix $D_2(\pi)$. 88
A.10 Created R_3 matrix (symmetric matrix). The dots can be determined by the condition that all the Q_3 rows add up to 0. 89
A.11 Created π_3. It determines the matrix $D_3(\pi)$. 89

List of Tables

1.1 An example of a multiple sequence alignment D of four homologous proteins from species *Persea americana*, *Cajanus cajan*, *Utricularia gibba* and *Vitis vinifera*. This data was provided by the Avocado Genome Consortium. 6

3.1 A multiple sequence alignment D of protein sequences of three sequences. 27

4.1 Proportion of correctly inferred trees by SVD from the alignments with length 1000 and that had evolved under the homogeneous JTT model. 43

4.2 Proportion of correctly inferred trees by SVD from the alignments with length 1000 and that had evolved under the homogeneous WAG model. 43

4.3 Proportion of correctly inferred trees by SVD from the alignments with length 1000 and that had evolved under the homogeneous LG model. 44

4.4 Proportion of correctly inferred trees by SVD from the three alignments concatenated. Each partition had a length of 1000 and had evolved under the homogeneous JTT, WAG or LG models. Results are shown as the alignments were considered as one partition and three partitions. 45

4.5 Average of computational time for SVD and IQ-TREE on one alignment (under the same hardware conditions). Results are shown depending on the alignment length and the considered partitions. . . 46

4.6 Abbreviations and phylogenetic groups of the 19 considered species. Notice that *Persea americana* is a dicotyledon based on its morphology, but its phylogenetic group is unknown. 48

4.7 Inferred rate matrices from the 10 selected protein alignments assuming a heterogeneous model across lineages with at most three different rate matrices. 53

4.8 Three scale factors for each (row) partition that were multiplied to the estimated branch lengths of the Protein 1. 54

4.9 Average of computational time on one alignment for the three main reconstruction strategies (same hardware conditions). Results are shown depending on whether a single partition or more has been considered (in case of the SVD+WO, the results correspond to the average of one alignment treating it like $r = 4, 12$ and 19, and $r = 189$ in case of the IQ-TREE). 65
4.10 Average of computational time on one alignment for SVD and IQ-TREE (same hardware conditions). Results are shown depending on alignment length and the partitions considered (in case of the SVD+WO, the results correspond to the average of one alignment treating it like $r = 4, 12$ and 19, and $r = 189$ in case of the IQ-TREE). 66

B.1 The genetic code shows the amino-acid that is encoded by each possible codon. Notice the redundancy in the code: most of the amino acids being encoded by more than one codon. Taken from [HA05] 91
B.2 Empirical amino-acid exchange rate matrices that IQ-TREE considers in its computations (alphabetical order). 92
B.3 Number of CPUs and RAM of the different nodes of hardware Hercules. It belongs to EGB group at the UB. 93
B.4 The rate matrices inferred from the 189 protein alignments of real data, assuming a heterogeneous model across lineages with at most three different rate matrices, and their length. 98

C.1 The main software packages developed in order to perform the thesis. 108

References

[AR03] E.S. Allman and J.A. Rhodes. Phylogenetic invariants for the general markov model of sequence mutation. *Mathematical Biosciences*, 186(2):113–144, 2003.

[AR04] E.S. Allman and J.A. Rhodes. *Mathematical Models in Biology: An Introduction*. Cambridge University Press, January 2004.

[AR05] E.S. Allman and J.A. Rhodes. The mathematics of phylogenetics. University of Alaska Fairbanks, 2005.

[AR08] E.S. Allman and J.A. Rhodes. Phylogenetic ideals and varieties for the general markov model. *Advances in Applied Mathematics*, 40(2):127–148, 2008.

[AWMH00] J. Adachi, P. Waddell, W. Martin, and M. Hasegawa. Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast dna. *Journal of Molecular Evolution*, 50(4):348–358, 2000.

[AZP05] F. Abascal, R. Zardoya, and D. Posada. Prottest: selection of best-fit models of protein evolution. *Bioinformatics*, 21(5):2104–2105, 2005.

[Bun71] P. Buneman. The recovery of trees from measures of dissimilarity. In D.G. Kendall and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1st edition, 1971.

[Cas12] M Casanellas. Algebraic tools for evolutionary biology. *La Gaceta de la RSME*, 15:521–536, 2012.

[CFS10] M. Casanellas and J. Fernández-Sánchez. Reconstrucción filogenética usando geometría algebraica. *Arbor. Ciencia, pensamiento, cultura*, 186, No 746:207–229, 2010.

[CFS16] M. Casanellas and J. Fernández-Sánchez. Invariant versus classical quartet inference when evolution is heterogeneous across sites and lineages. *Systematic Biology*, 65(2):280–291, 2016.

[DLGL10] C.C. Dang, Q.S. Le, O. Gascuel, and V.S. Le. Flu, an amino acid substitution model for influenza proteins. *BMC Evolutionary Biology*, 10(1):99–100, 2010.

[DSO78] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5, suppl. 3:345–352, 1978.

[FM67] W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155(3760):279–284, 1967.

[G+13] L. Guéguen et al. Bio++: Efficient extensible libraries and tools for computational molecular evolution. *Molecular Biology and Evolution*, 30(8):1745–1750, 2013.

[Gas05] O. Gascuel. *Mathematics of Evolution and Phylogeny*. Oxford University Press, 2005.

[GL96] G.H. Golub and C.F.V. Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[GXL08] N. Glansdorff, Y. Xu, and B. Labedan. The last universal common ancestor: emergence, constitution and genetic legacy of an elusive forerunner. *Biology Direct*, Volume 3(29):127–136, 2008.

[HA05] P.G. Higgs and T. Attwood. In *Bioinformatics and Molecular Evolution*, chapter 2. Blackwell Publishing, 2005.

[HH92] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89(22):10915–10919, 1992.

[JTT92] D.T. Jones, W.R. Taylor, and J.M. Thornton. The rapid generation of mutation data matrices from protein sequences. *Bioinformatics*, 8(3):275–282, 1992.

[KG05] C. Kosiol and N. Goldman. Different versions of the Dayhoff rate matrix. *Molecular Biology and Evolution*, 22(2):193–199, 2005.

[Koo03] EV. Koonin. Comparative genomics, minimal gene-sets and the last universal common ancestor. *Nature Reviews Microbioly*, Volume 1(2):127–136, 2003.

[LG08] S.Q. Le and O. Gascuel. An improved general amino acid replacement matrix. *Molecular Biology and Evolution*, 25(7):1307–1320, 2008.

[LSV09] P. Lemey, M. Salemi, and A. Vandemme. In *The Phylogenetic Handbook*, pages 123–125. Cambridge University Press, 2nd edition, 2009.

[MM81] T. Margush and F.R. McMorris. Consensusn-trees. *Bulletin of Mathematical Biology*, 43(2):239–244, 1981.

[NSHM15] L-T. Nguyen, H.A. Schmidt, A. von Haeseler, and B.Q. Minh. Iq-tree: A fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular biology and evolution*, 32(1):268–274, 2015.

[OR07] T. H. Ogden and M. S. Rosenberg. How should gaps be treated in parsimony? A comparison of approaches using simulation. *Molecular Phylogenetics and Evolution*, 42:817–826, 2007.

[Pev15] J. Pevsner. In *Bioinformatics and Functional Genomics*, chapter 1. Wiley Blackwell, 3rd edition, 2015.

[PFTV92] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. Root finding and nonlinear sets of equations. In *Numerical Recipes in C: The Art of Scientific Computing*, chapter 9. Cambridge University Press, 2nd edition, 1992.

[PS05] L. Pachter and B. Sturmfels. In *Algebraic Statistics for computational biology*. Cambridge University Press, November 2005.

[RG97] A. Rambaut and N.C. Grass. Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Bioinformatics*, 13(3):235–238, 1997.

[RG01a] V. Ranwez and O. Gascuel. Phylogenetic reconstruction algorithms based on weighted 4-trees. In O. Gascuel and M-F. Sagot, editors, *Computational biology*, volume 2066, page 84–98. Springer, 2001.

[RG01b] V. Ranwez and O. Gascuel. Quartet-based phylogenetic inference: Improvements and limits. *Molecular Biology and Evolution*, 18(6):1103–1116, 2001.

[Sab14] M. Sabaté. Reconstruction of phylogenetic trees using quartet methods. Master's thesis, Universitat Politècnica de Catalunya, Barcelona, 2014.

[SC16] C. Sanderson and R. Curtin. Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, 2016.

[SH10] J. Sukumaran and M.T. Holder. Dendropy: A python library for phylogenetic computing. *Bioinformatics*, 26:1569–1571, 2010.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[SOWH96] D. Swofford, G.J. Olsen, P.J. Waddell, and D.M Hillis. Phylogenetic inference. In C. Moritz D. Hillis and B. Mable, editors, *Molecular Systematics*, chapter 11. Cambridge University Press, 2nd edition, 1996.

[Ste93] G.W. Stewart. On the early history of the singular value decomposition. *SIAM Rev.*, Volume 35(4):551–566, December 1993.

[Tav86] S. Tavaré. Some probabilistic and statistical problems in the analysis of dna sequences. *Lectures on mathematics in the life sciences*, Volume 17(2):57–86, 1986.

[VST03] S. Veerassamy, A. Smith, and E.R.M. Tillier. A transition probability model for amino acid substitutions from blocks. *Journal of Computational Biology*, 10(6):997–1010, 2003.

[WG01] S. Whelan and N. Goldman. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular Biology and Evolution*, 18(5):691–699, 2001.

[Yan96] Z. Yang. Among-site rate variation and its impact on phylogenetic analyses. *Trends in Ecology and Evolution*, 11:367–372, 1996.

Appendices

A Auxiliary Figures

A.1 Chemical structure of amino-acids and their abbreviations

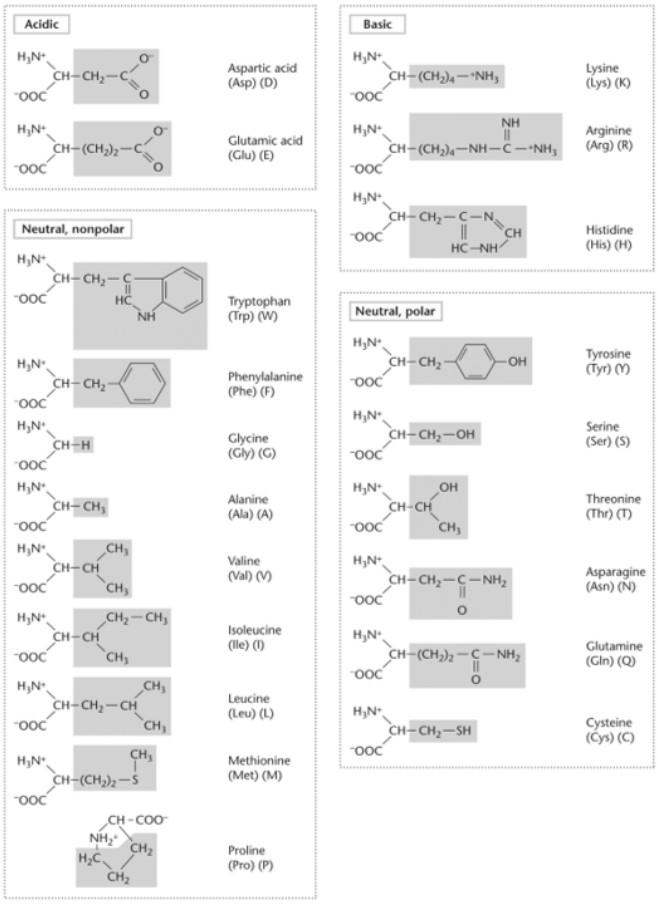


Figure A.1: Chemical structure of the 20 amino-acids. They are classified by the properties of their residue group R into four groups: acidic, basic, nonpolar and polar amino-acids. Taken from [HA05].

A.2 Matrix A: observable exchanges

$$
\mathrm{A} = \begin{array}{c} \\ \mathtt{A} \\ \mathtt{R} \\ \mathtt{N} \\ \mathtt{D} \\ \mathtt{C} \\ \mathtt{Q} \\ \mathtt{E} \\ \mathtt{G} \\ \mathtt{H} \\ \mathtt{I} \\ \mathtt{L} \\ \mathtt{K} \\ \mathtt{M} \\ \mathtt{F} \\ \mathtt{P} \\ \mathtt{S} \\ \mathtt{T} \\ \mathtt{W} \\ \mathtt{Y} \\ \mathtt{V} \end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccccccc} \mathtt{A} & \mathtt{R} & \mathtt{N} & \mathtt{D} & \mathtt{C} & \mathtt{Q} & \mathtt{E} & \mathtt{G} & \mathtt{H} & \mathtt{I} & \mathtt{L} & \mathtt{K} & \mathtt{M} & \mathtt{F} & \mathtt{P} & \mathtt{S} & \mathtt{T} & \mathtt{W} & \mathtt{Y} & \mathtt{V} \end{array} \\
\left[\begin{array}{cccccccccccccccccccc}
0 & & & & & & & & & & & & & & & & & & & \\
30 & 0 & & & & & & & & & & & & & & & & & & \\
109 & 17 & 0 & & & & & & & & & & & & & & & & & \\
154 & 0 & 532 & 0 & & & & & & & & & & & & & & & & \\
33 & 10 & 0 & 0 & 0 & & & & & & & & & & & & & & & \\
93 & 120 & 50 & 76 & 0 & 0 & & & & & & & & & & & & & & \\
265 & 0 & 94 & 831 & 0 & 422 & 0 & & & & & & & & & & & & & \\
579 & 10 & 156 & 162 & 10 & 30 & 112 & 0 & & & & & & & & & & & & \\
21 & 103 & 226 & 43 & 10 & 243 & 23 & 10 & 0 & & & & & & & & & & & \\
66 & 30 & 36 & 13 & 17 & 8 & 35 & 0 & 3 & 0 & & & & & & & & & & \\
95 & 17 & 37 & 0 & 0 & 75 & 15 & 17 & 40 & 253 & 0 & & & & & & & & & \\
57 & 477 & 322 & 85 & 0 & 147 & 104 & 60 & 23 & 43 & 39 & 0 & & & & & & & & \\
29 & 17 & 0 & 0 & 0 & 20 & 7 & 7 & 0 & 57 & 207 & 90 & 0 & & & & & & & \\
20 & 7 & 7 & 0 & 0 & 0 & 0 & 17 & 20 & 90 & 167 & 0 & 17 & 0 & & & & & & \\
345 & 67 & 27 & 10 & 10 & 93 & 40 & 49 & 50 & 7 & 43 & 43 & 4 & 7 & 0 & & & & & \\
772 & 137 & 432 & 98 & 117 & 47 & 86 & 450 & 26 & 20 & 32 & 168 & 20 & 40 & 269 & 0 & & & & \\
590 & 20 & 169 & 57 & 10 & 37 & 31 & 50 & 14 & 129 & 52 & 200 & 28 & 10 & 73 & 696 & 0 & & & \\
0 & 27 & 3 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 13 & 0 & 0 & 10 & 0 & 17 & 0 & 0 & & \\
20 & 3 & 36 & 0 & 30 & 0 & 10 & 0 & 40 & 13 & 23 & 10 & 0 & 260 & 0 & 22 & 23 & 6 & 0 & \\
365 & 20 & 13 & 17 & 33 & 27 & 37 & 97 & 30 & 661 & 303 & 17 & 77 & 10 & 50 & 43 & 186 & 0 & 17 & 0
\end{array}\right]
\end{array}
$$

Figure A.2: Number of accepted point mutations (multiplied by 10) accumulated from closely related sequences (symmetric matrix). Taken from [DSO78].

A.3 Estimated state distribution at $\mathbf{X}_r$ (PAM models)

$$
\pi^r = \begin{array}{c} \\ \mathtt{A} \\ \mathtt{R} \\ \mathtt{N} \\ \mathtt{D} \\ \mathtt{C} \\ \mathtt{Q} \\ \mathtt{E} \\ \mathtt{G} \\ \mathtt{H} \\ \mathtt{I} \\ \vdots \end{array}
\begin{array}{c}
\left[\begin{array}{c} 0.087 \\ 0.041 \\ 0.040 \\ 0.047 \\ 0.033 \\ 0.038 \\ 0.050 \\ 0.089 \\ 0.034 \\ 0.037 \\ \vdots \end{array}\right]
\end{array}
\quad
\begin{array}{c} \vdots \\ \mathtt{L} \\ \mathtt{K} \\ \mathtt{M} \\ \mathtt{F} \\ \mathtt{P} \\ \mathtt{S} \\ \mathtt{T} \\ \mathtt{W} \\ \mathtt{Y} \\ \mathtt{V} \end{array}
\begin{array}{c}
\left[\begin{array}{c} \vdots \\ 0.085 \\ 0.081 \\ 0.015 \\ 0.040 \\ 0.051 \\ 0.070 \\ 0.058 \\ 0.010 \\ 0.030 \\ 0.065 \end{array}\right]
\end{array}
$$

Figure A.3: Relative frequencies of the amino-acids in the Accepted Point Mutation Data. Taken from [DSO78].

A.4 PAM1

PAM1 =

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

Figure A.4: Transition matrix for the evolutionary distance of 1 PAM. To simplify its appearance, the entries are shown multiplied by 10 000. Taken from [DSO78].

A.5 PAM250

PAM250 =

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V	7	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	7	2	4	17

Figure A.5: Transition matrix for the evolutionary distance of 250 PAMs. To simplify its appearance, the entries are shown multiplied by 100. Taken from [DSO78].

A.6 R_1

$R_1 =$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	·																			
R	1	·																		
N	2	3	·																	
D	4	5	6	·																
C	7	8	9	10	·															
Q	11	12	13	14	15	·														
E	16	17	18	19	20	21	·													
G	22	23	24	25	26	27	28	·												
H	29	30	31	32	33	34	35	36	·											
I	37	38	39	40	41	42	43	44	45	·										
L	46	47	48	49	50	51	52	53	54	55	·									
K	56	57	58	59	60	61	62	63	64	65	66	·								
M	67	68	69	70	71	72	73	74	75	76	77	78	·							
F	79	80	81	82	83	84	85	86	87	88	89	90	91	·						
P	92	93	94	95	96	97	98	99	100	101	102	103	104	105	·					
S	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	·				
T	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	·			
W	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	·		
Y	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	·	
V	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	·

Figure A.6: Created R_1 matrix (symmetric matrix). The dots can be determined by the condition that all the Q_1 rows add up to 0.

A.7 State distribution at X_r (Q_1 model)

$\pi_1 =$

A	0.091904	⋮	⋮
R	0.061830	L	0.040126
N	0.058676	K	0.076748
D	0.073152	M	0.040752
C	0.068765	F	0.023826
Q	0.066004	P	0.014261
E	0.051691	S	0.050901
G	0.051544	T	0.019803
H	0.022944	W	0.058565
I	0.042645	Y	0.032102
⋮	⋮	V	0.053761

Figure A.7: Created π_1. It determines the matrix $D_1(\pi)$.

A.8 R_2

$R_2 =$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	·																			
R	646	·																		
N	10	81	·																	
D	119	45	7	·																
C	59	29	115	4	·															
Q	528	5	378	35	42	·														
E	477	18	12	15	73	15	·													
G	113	46	29	328	54	5	10	·												
H	475	12	180	24	21	8	23	5	·											
I	52	115	285	58	56	7	62	179	18	·										
L	26	55	391	54	43	70	17	27	292	16	·									
K	101	57	103	112	25	12	64	33	53	51	4	·								
M	49	23	597	201	14	74	62	137	58	59	38	40	·							
F	33	248	9	20	479	38	26	8	53	323	18	245	27	·						
P	59	164	16	4	16	181	503	12	14	20	40	118	9	89	·					
S	961	263	47	8	24	15	194	22	46	21	15	11	5	45	81	·				
T	9	45	69	34	24	32	16	65	14	767	26	10	86	38	72	92	·			
W	112	9	226	229	11	9	63	4	14	232	36	209	536	47	71	573	78	·		
Y	31	298	31	6	8	6	24	43	44	17	6	130	16	102	10	25	7	10	·	
V	47	17	126	323	56	11	223	30	388	23	18	310	21	105	30	32	30	10	35	·

Figure A.8: Created R_2 matrix (symmetric matrix). The dots can be determined by the condition that all the Q_2 rows add up to 0.

A.9 State distribution at X_r (Q_2 model)

$\pi_2 =$

A	0.019	⋮	⋮
R	0.051	L	0.072
N	0.03	K	0.021
D	0.056	M	0.085
C	0.045	F	0.069
Q	0.01	P	0.025
E	0.082	S	0.037
G	0.027	T	0.019
H	0.02	W	0.081
I	0.028	Y	0.066
⋮	⋮	V	0.157

Figure A.9: Created π_2. It determines the matrix $D_2(\pi)$.

A.10 R_3

$$
R_3 = \begin{bmatrix}
\cdot & & & & & & & & & & & & & & & & & & & \\
7.2 & \cdot & & & & & & & & & & & & & & & & & & \\
94.6 & 9 & \cdot & & & & & & & & & & & & & & & & & \\
34.5 & 7.1 & 4.1 & \cdot & & & & & & & & & & & & & & & & \\
90.3 & 3.5 & 644.9 & 17.1 & \cdot & & & & & & & & & & & & & & & \\
39.4 & 0.2 & 4.5 & 6.7 & 11.8 & \cdot & & & & & & & & & & & & & & \\
79.8 & 12.9 & 29 & 167.8 & 334 & 1.5 & \cdot & & & & & & & & & & & & & \\
7.4 & 222.3 & 56.9 & 23.7 & 16.3 & 305.1 & 279.6 & \cdot & & & & & & & & & & & & \\
33.2 & 1.6 & 3.3 & 257.1 & 7.3 & 10 & 52.2 & 96.4 & \cdot & & & & & & & & & & & \\
272.5 & 7.1 & 26 & 22.6 & 3.2 & 83 & 61.7 & 163.2 & 174 & \cdot & & & & & & & & & & \\
3.2 & 38.4 & 52.3 & 0.2 & 23.9 & 27.2 & 26.5 & 10.1 & 2.3 & 23 & \cdot & & & & & & & & & \\
6.6 & 13.8 & 59.7 & 20.6 & 15.6 & 420.2 & 19.3 & 0.2 & 23.1 & 449.7 & 636.3 & \cdot & & & & & & & & \\
421.8 & 9.7 & 1 & 50 & 45 & 7.9 & 33.6 & 19.4 & 8.1 & 3.1 & 76.7 & 6.1 & \cdot & & & & & & & \\
10.3 & 4.2 & 6.4 & 79.7 & 396.4 & 4.3 & 8.4 & 10.5 & 5.4 & 2.3 & 0.2 & 9.5 & 2.6 & \cdot & & & & & & \\
24.3 & 2.1 & 215.6 & 6.6 & 82.4 & 184.6 & 15.2 & 5.7 & 75.5 & 13.1 & 378.1 & 243.8 & 883.6 & 11.1 & \cdot & & & & & \\
3.4 & 590.5 & 413.4 & 6.4 & 14.9 & 0.2 & 1.7 & 41.4 & 17.9 & 3.4 & 1710.6 & 199 & 321.9 & 234.2 & 61.1 & \cdot & & & & \\
48.7 & 49.1 & 16.9 & 3.5 & 9.5 & 3.7 & 32.2 & 425.2 & 1.3 & 3.8 & 0.5 & 0.2 & 8.9 & 5.2 & 13.1 & 51.4 & \cdot & & & \\
52.2 & 29.7 & 666.3 & 144.9 & 12.3 & 5.1 & 16.1 & 3.6 & 2 & 7.2 & 1.3 & 0.3 & 16.3 & 73.7 & 43.1 & 7.3 & 656.3 & \cdot & & \\
126.1 & 0.2 & 7.5 & 13.7 & 49.5 & 289.3 & 190.3 & 24.3 & 61.8 & 95.3 & 141.6 & 36.1 & 15.6 & 349.4 & 19.3 & 617 & 37.5 & 59.6 & \cdot & \\
15.6 & 266.5 & 0.2 & 0.2 & 0.2 & 292.3 & 83.9 & 0.2 & 228.9 & 136.5 & 99.3 & 58.2 & 231 & 48.8 & 264.8 & 5.8 & 11.4 & 13.9 & 55.6 & \cdot
\end{bmatrix}
$$

Figure A.10: Created R_3 matrix (symmetric matrix). The dots can be determined by the condition that all the Q_3 rows add up to 0.

A.11 State distribution at X_r (Q_3 model)

$$
\pi_3 = \begin{bmatrix}
\texttt{A} & 0.066004 \\
\texttt{R} & 0.051691 \\
\texttt{N} & 0.019803 \\
\texttt{D} & 0.058676 \\
\texttt{C} & 0.076748 \\
\texttt{Q} & 0.050901 \\
\texttt{E} & 0.058565 \\
\texttt{G} & 0.022944 \\
\texttt{H} & 0.032102 \\
\texttt{I} & 0.051544 \\
\vdots & \vdots
\end{bmatrix}
\quad
\begin{bmatrix}
\vdots & \vdots \\
\texttt{L} & 0.042645 \\
\texttt{K} & 0.040752 \\
\texttt{M} & 0.014261 \\
\texttt{F} & 0.091904 \\
\texttt{P} & 0.053761 \\
\texttt{S} & 0.073152 \\
\texttt{T} & 0.068765 \\
\texttt{W} & 0.040126 \\
\texttt{Y} & 0.06183 \\
\texttt{V} & 0.023826
\end{bmatrix}
$$

Figure A.11: Created π_3. It determines the matrix $D_3(\pi)$.

B
Auxiliary Tables

B.1 The genetic code

First position	Second position				Third position
	U	C	A	G	
U	Phe (F) Phe (F) Leu (L) Leu (L)	Ser (S) Ser (S) Ser (S) Ser (S)	Tyr (Y) Tyr (Y) STOP STOP	Cys (C) Cys (C) STOP Trp (W)	U C A G
C	Leu (L) Leu (L) Leu (L) Leu (L)	Pro (P) Pro (P) Pro (P) Pro (P)	His (H) His (H) Gln (Q) Gln (Q)	Arg (R) Arg (R) Arg (R) Arg (R)	U C A G
A	Ile (I) Ile (I) Ile (I) Met (M)	Thr (T) Thr (T) Thr (T) Thr (T)	Asn (N) Asn (N) Lys (K) Lys (K)	Ser (S) Ser (S) Arg (R) Arg (R)	U C A G
G	Val (V) Val (V) Val (V) Val (V)	Ala (A) Ala (A) Ala (A) Ala (A)	Asp (D) Asp (D) Glu (E) Glu (E)	Gly (G) Gly (G) Gly (G) Gly (G)	U C A G

Table B.1: The genetic code shows the amino-acid that is encoded by each possible codon. Notice the redundancy in the code: most of the amino acids being encoded by more than one codon. Taken from [HA05]

B.2 Amino-acid rate matrices supported by IQ-TREE

Model	Basic description	Authors	Year
BLOSUM62	BLOcks SUbstitution Matrix	Henikoff and Henikoff	1992
cpREV	Chloroplast matrix	Adachi et al.	2000
PAM	General matrix	Dayhoff et al.	1978
PAMDCMut	Revised Dayhoff matrix	Kosiol and Goldman	2005
FLU	Influenza virus	Dang et al.	2010
HIVb	HIV between-patient matrix HIV-Bm	Nickle et al.	2007
HIVw	HIV within-patient matrix HIV-Wm	Nickle et al.	2007
JTT	General matrix	Jones et al.	1992
JTTDCMut	Revised JTT matrix	Kosiol and Goldman	2005
LG	General matrix	Le and Gascuel	2008
mtART	Mitochondrial Arthropoda	Abascal et al.	2007
mtMAM	Mitochondrial Mammalia	Yang et al.	1998
mtREV	Mitochondrial Verterbrate	Adachi and Hasegawa	1996
mtZOA	Mitochondrial Metazoa (Animals)	Rota-Stabelli et al.	2009
mtMet	Mitochondrial Metazoa	Vinh et al.	2017
mtVer	Mitochondrial Verterbrate	Vinh et al.	2017
mtInv	Mitochondrial Inverterbrate	Vinh et al.	2017
Poisson	Equal amino-acid exchange rates and frequencies	-	-
PMB	Probability Matrix from Blocks, revised BLOSUM matrix	Veerassamy et al.	2004
rtREV	Retrovirus	Dimmic et al.	2002
VT	General matrix	Mueller and Vingron	2000
WAG	General matrix	Whelan and Goldman	2001

Table B.2: Empirical amino-acid exchange rate matrices that IQ-TREE considers in its computations (alphabetical order).

B.3 Computational capability of Hercules server

	Name	CPUs	RAM
Principal node	Herculesd1	8	64G
Computional nodes	hercules01	8	7.8G
	hercules02	8	7.8G
	hercules03	8	7.8G
	hercules04	8	7.8G
	hercules05	8	15.7G
	hercules06	8	31.5G
	hercules07	8	62.1G
	hercules08	24	126.2G
	hercules09	24	126.2G
	hercules10	24	252.4G
	hercules11	64	757.4G
	hercules12	64	1007.9G
Backup node	Herculesd0	128	8G

Table B.3: Number of CPUs and RAM of the different nodes of hardware Hercules. It belongs to EGB group at the UB.

B.4 Estimated rate matrices and alignment length for each protein of real data

Protein	Q for edges joining monocotyledons	Q for edges joining dicotyledons	Q for the rest of the edges	Alignment length
1	JTT	JTT	JTT	580
2	JTT	JTT	LG	370
3	JTT	JTT	JTT	913
4	JTT	JTT	JTT	535
5	LG	JTT	rtREV	381
6	JTT	JTTDCMut	VT	457
7	JTT	JTT	JTT	479
8	JTT	JTT	VT	344
9	JTT	JTT	cpREV	282
10	JTT	JTT	JTT	622
11	JTT	JTT	JTT	472
12	JTT	JTT	cpREV	384
13	JTT	JTT	JTT	546
14	JTT	JTT	JTT	903
15	LG	LG	rtREV	448
16	JTT	JTT	FLU	465
17	JTT	JTTDCMut	JTT	373
18	JTTDCMut	JTTDCMut	WAG	230
19	JTT	JTT	JTTDCMut	712
20	JTT	JTT	JTT	464
21	JTT	JTT	LG	252
22	JTT	LG	LG	396
23	LG	LG	LG	354
24	JTT	JTT	JTT	2061
25	JTTDCMut	JTT	VT	339
26	JTT	JTT	LG	484
27	JTT	JTT	WAG	245
28	JTT	JTT	JTT	340
29	JTTDCMut	JTTDCMut	WAG	453
30	JTT	JTT	JTT	359
31	JTTDCMut	JTT	JTT	248
32	JTT	JTT	JTT	397
33	JTT	JTT	JTT	316
34	JTT	LG	VT	205
35	JTT	cpREV	cpREV	594
36	JTT	JTT	JTT	308
37	JTT	JTT	LG	610
38	JTT	JTT	cpREV	547
39	JTT	JTT	LG	228
40	JTT	LG	JTT	553

Protein	Q for edges joining monocotyledons	Q for edges joining dicotyledons	Q for the rest of the edges	Alignment length
41	JTTDCMut	JTTDCMut	WAG	597
42	cpREV	cpREV	cpREV	299
43	JTT	cpREV	FLU	443
44	JTT	JTT	JTTDCMut	400
45	JTT	JTTDCMut	JTT	507
46	JTT	JTT	JTT	402
47	JTT	JTT	JTT	359
48	JTT	JTT	JTT	169
49	JTT	JTT	JTT	269
50	JTTDCMut	LG	WAG	289
51	JTT	JTT	JTT	610
52	JTT	JTT	JTT	611
53	JTT	JTT	JTT	530
54	JTT	JTT	LG	302
55	JTTDCMut	JTT	cpREV	343
56	JTT	JTTDCMut	JTT	303
57	JTT	JTTDCMut	LG	433
58	JTT	JTT	JTT	201
59	JTT	JTTDCMut	JTTDCMut	606
60	JTT	JTTDCMut	JTTDCMut	120
61	JTT	JTT	cpREV	303
62	JTT	JTT	JTT	245
63	JTT	JTT	JTT	796
64	JTT	JTT	VT	244
65	JTT	JTT	JTT	344
66	JTT	JTT	JTT	782
67	JTT	JTT	JTT	628
68	JTT	JTT	VT	246
69	JTT	JTTDCMut	JTT	382
70	JTT	JTT	JTT	911
71	JTT	LG	cpREV	168
72	LG	LG	LG	431
73	JTT	JTT	JTTDCMut	284
74	LG	LG	LG	124
75	JTTDCMut	JTTDCMut	VT	641
76	LG	JTT	LG	293
77	JTT	JTT	VT	336
78	JTTDCMut	JTT	JTT	362
79	JTTDCMut	LG	LG	531
80	JTT	JTT	JTT	933

Protein	Q for edges joining monocotyledons	Q for edges joining dicotyledons	Q for the rest of the edges	Alignment length
81	JTT	JTT	JTT	394
82	JTT	JTT	JTT	184
83	JTT	JTT	VT	176
84	JTT	JTT	JTT	411
85	LG	LG	LG	351
86	JTT	JTTDCMut	WAG	233
87	JTT	JTT	VT	333
88	LG	LG	JTTDCMut	469
89	JTT	JTT	JTT	538
90	JTT	JTT	LG	688
91	JTT	JTT	JTT	663
92	JTT	cpREV	cpREV	222
93	JTT	JTT	JTT	506
94	LG	cpREV	JTTDCMut	351
95	JTT	JTT	LG	352
96	JTTDCMut	JTT	JTT	900
97	JTT	JTT	LG	1364
98	JTTDCMut	JTT	WAG	622
99	JTT	JTT	LG	415
100	JTT	JTT	JTT	337
101	JTT	JTT	JTT	561
102	JTT	cpREV	cpREV	252
103	JTTDCMut	JTTDCMut	LG	225
104	JTT	JTT	JTT	643
105	WAG	JTT	cpREV	308
106	JTT	JTTDCMut	JTT	672
107	JTTDCMut	JTTDCMut	JTTDCMut	719
108	cpREV	cpREV	cpREV	467
109	JTT	JTTDCMut	JTT	242
110	JTT	JTTDCMut	JTTDCMut	504
111	JTT	JTTDCMut	JTT	278
112	JTT	JTT	JTT	679
113	JTT	JTT	JTT	486
114	JTT	JTT	JTT	446
115	JTT	LG	cpREV	213
116	JTT	JTT	WAG	447
117	JTT	JTT	VT	316
118	JTT	JTTDCMut	JTTDCMut	397
119	JTT	JTT	JTT	294
120	JTT	JTT	JTT	1201

Protein	Q for edges joining monocotyledons	Q for edges joining dicotyledons	Q for the rest of the edges	Alignment length
121	JTTDCMut	JTT	JTT	333
122	JTT	JTT	JTT	494
123	JTT	JTT	VT	279
124	JTT	LG	VT	160
125	JTT	JTTDCMut	WAG	166
126	JTT	JTT	JTTDCMut	1692
127	mtInv	mtMet	mtZOA	300
128	WAG	JTTDCMut	WAG	183
129	JTT	JTT	VT	244
130	JTT	JTT	JTTDCMut	273
131	JTT	JTT	JTT	620
132	JTT	JTT	JTT	582
133	JTTDCMut	WAG	cpREV	342
134	JTTDCMut	JTTDCMut	JTTDCMut	377
135	LG	LG	VT	615
136	LG	LG	VT	302
137	cpREV	LG	WAG	249
138	JTTDCMut	JTT	JTTDCMut	393
139	JTT	JTT	JTT	913
140	JTT	LG	cpREV	371
141	JTT	JTT	JTT	516
142	JTT	JTTDCMut	JTT	1095
143	JTT	JTTDCMut	JTT	550
144	JTTDCMut	JTT	JTT	375
145	JTT	JTT	JTT	311
146	JTT	JTT	JTTDCMut	386
147	JTT	JTT	JTTDCMut	453
148	JTT	JTTDCMut	LG	378
149	JTT	JTT	JTT	201
150	JTT	JTT	JTT	440
151	WAG	JTTDCMut	PAMDCMut	257
152	JTT	JTT	cpREV	183
153	WAG	JTT	JTT	304
154	JTT	JTT	JTT	644
155	JTT	JTT	LG	173
156	JTT	LG	LG	159
157	JTT	JTT	JTT	839
158	JTT	JTT	JTT	679
159	JTT	JTT	JTT	512
160	JTT	JTT	JTT	1057

Protein	Q for edges joining monocotyledons	Q for edges joining dicotyledons	Q for the rest of the edges	Alignment length
161	JTT	JTT	JTT	403
162	JTT	JTT	cpREV	958
163	JTT	JTT	JTT	721
164	JTT	JTT	JTTDCMut	473
165	JTT	LG	JTTDCMut	364
166	JTT	JTT	VT	324
167	JTT	JTT	WAG	859
168	JTT	JTT	JTT	264
169	JTT	JTT	JTT	602
170	JTT	JTT	JTT	882
171	JTT	JTT	LG	293
172	JTT	JTT	VT	583
173	JTT	JTT	JTT	650
174	JTT	JTT	VT	263
175	JTT	JTTDCMut	WAG	324
176	JTT	JTT	JTTDCMut	448
177	JTT	JTTDCMut	JTT	190
178	JTT	JTT	JTT	667
179	JTT	LG	VT	119
180	JTT	JTT	JTT	271
181	JTT	JTT	JTT	512
182	JTT	JTTDCMut	LG	414
183	cpREV	cpREV	cpREV	329
184	JTT	JTT	LG	644
185	JTT	JTT	WAG	267
186	WAG	JTTDCMut	LG	229
187	JTT	JTT	VT	408
188	JTT	JTT	LG	295
189	JTT	VT	VT	203

Table B.4: The rate matrices inferred from the 189 protein alignments of real data, assuming a heterogeneous model across lineages with at most three different rate matrices, and their length.

C Implemented software

C.1 Main code of SVD

```
/*####################################
#                TFG 2018            #
#             Code: SVD.cpp          #
####################################*/

#include "Clean_declaration.h"
#include <stdlib.h>
#include <stdio.h>
#include <algorithm>
#include <armadillo>
#include <string>
#include <fstream>
#include <iostream>
#include <time.h>
#include <unistd.h>

using namespace std;
using namespace arma; // for 'armadillo'

int main(int argc, char *argv[]) {

  string filename = argv[1];
  // Characters that do not belong to the alphabet, i.e.,
   gaps
  string chr_exclude = argv[2];
  int nb_partitions = stoi(argv[3]);

  double dist_F1,dist_F2,dist_F3;

  Alignment align = readFASTA(filename);
  // Vector of joint distribution
  map<string,int> tensor = getColumns(align,chr_exclude);
```

```
// Construct the alphabet from the data
map<char,int> alph = constructAlph(align,chr_exclude);
int nb_alph = alph.size();

// Sort the name of taxa by the order of appearance of
 the fasta file
vector<int> order_taxa = realOrder(align.taxa);
vector<int> split_1, split_2;

// Flattening matrix for 12|34 and its Frobeniusdistance
  to E_{rk}
split_1 = realSplit({0,1},order_taxa);
split_2 = realSplit({2,3},order_taxa);
mat flat_01 = flatteningMat(tensor,alph,split_1,split_2,
  align.seq_len);
dist_F1 = distFrobenius(flat_01,nb_partitions,nb_alph);

// Flattening matrix 13|24 and its Frobenius distance to
  E_{rk}
split_1 = realSplit({0,2},order_taxa);
split_2 = realSplit({1,3},order_taxa);
mat flat_02 = flatteningMat(tensor,alph,split_1,split_2,
 align.seq_len);
dist_F2 = distFrobenius(flat_02,nb_partitions,nb_alph);

// Flattening matrix 14|23 and its Frobenius distance to
  E_{rk}
split_1 = realSplit({0,3},order_taxa);
split_2 = realSplit({1,2},order_taxa);
mat flat_03 = flatteningMat(tensor,alph,split_1,split_2,
  align.seq_len);
dist_F3 = distFrobenius(flat_03,nb_partitions,nb_alph);

double min_score = min(dist_F1, dist_F2);
min_score = min(min_score, dist_F3);

cout << endl;
cout << "Taxa 1: " << align.taxa[0] << endl;
cout << "Taxa 2: " << align.taxa[1] << endl;
cout << "Taxa 3: " << align.taxa[2] << endl;
cout << "Taxa 4: " << align.taxa[3] << endl;
cout << endl;
cout << "Topologies: \t" <<  "12|34 \t \t" << " 13|24 \t
  \t" << " 14|23" << endl;
cout << "Frobenius distance: \t" << dist_F1 << "\t \t"
 << dist_F2 << "\t \t" <<  dist_F3 << endl;
cout << endl;
```

```
  cout << "Inferred topology: \t";
  if(min_score == dist_F1) cout << " 12|34" << endl;
  else if(min_score == dist_F2) cout << " 13|24" << endl;
  else cout << " 23|14" << endl;
  cout << endl;

  return(0);
}
```

C.2 Functions of SVD

C.2.1 readFASTA

It reads an alignment to obtain the so-called Alignment struct.

```
struct Alignment {
  unsigned int num_taxa;
  unsigned int seq_len;
  vector<string> taxa; // the taxa names
  map<string,string> seqs;  // dictionary: taxa name -
   sequence
};

Alignment readFASTA (string fname) {

  Alignment align;
  string line;
  fstream file;
  string speciesname;
  string seq = "";
  int numchar;
  vector<int> al_length;
  // Opening file
  file.open(fname.c_str(), fstream::in);
  if (file == NULL) {
    cout << "cannot open file " << fname.c_str() << "\n";
    exit(1);
  }
  // Get the first line from the FASTA file
  getline(file, line, '\n');
  if (line[0]!='>'){
    cout << "error: not a FASTA file \n";
    exit(0);
  }
  numchar = line.length()-1;
```

```
  speciesname = line.substr(1,numchar);
  align.taxa.push_back(speciesname);


  while(! file.eof()) {
    getline(file, line, '\n');

    // Storage all the lines until it finds the next name
   specie
    if (line[0]! = '>') seq += line;
    else {
      align.seqs[speciesname]=seq;
      al_length.push_back(dna.length());
      numchar = line.length()-1;
      speciesname = line.substr(1,numchar);
      align.taxa.push_back(speciesname);
      seq.clear();
      if (align.seqs[speciesname].length() > 0)
        cerr << "Warning - found 2+ sequences with same
   name " << speciesname << endl;
    }
  }

    align.seqs[speciesname] = seq;
    al_length.push_back(seq.length());
    file.close();

    align.num_taxa = align.taxa.size();
    if(al_length.size() != align.num_taxa)  cout << "error
    in getting the alignment";
    for (int i=0; i< al_length.size()-1; i++) {
      if (al_length[i] != al_length[i+1]) {
          cout << "sequence " << align.taxa[i+1] << "does
   not have same length \n";
          exit(0);
    }
  }
  align.seq_len = al_length.back();
  return align;
}
```

C.2.2 getColumns

From an alignment, it obtains the absolute frequencies of each column; it performs the joint distribution vector (which may not contain all possible patterns).

```
// Check that the considered character is not equal to one
    of those that the user wanted to exclude(usually gaps)
bool findChrExclude(string chr_exclude, char to_find) {
  int len = chr_exclude.size();
  for(int i=0; i<len; ++i) {
    if(chr_exclude[i]==to_find) return true;
  }
  return false;
}

map<string, int> getColumns(Alignment &align, string
    chr_exclude) {
  int al_length=align.seq_len;

  map<string,string> myseqs=align.seqs;
  map<string,string>::const_iterator pos;
  map<string, int> columnscount;

  // Loop for all columns of the alignment
  for (int i=0; i < al_length; i++) {
    bool exclude = false;
    string col="";
        // Loop for all the taxa sequences to capture each
    column
    for (pos = myseqs.begin(); pos != myseqs.end() and not
    exclude; ++pos) {
          col.push_back(pos->second[i]);
          exclude = findChrExclude(chr_exclude, pos->
   second[i]);
      }
    if(not exclude) columnscount[col]++;
  }
  return (columnscount);
}
```

C.2.3 constructAlph

It takes all the different characters of the sequence in order to determine the alphabet of the alignment. Note that the alignment can induce an alphabet with any size.

```
map<char,int> constructAlph(Alignment &align, string
    chr_exclude) {

  int value = 0;
  int al_length=align.seq_len;
  map <char,int> alph;
  map<string,string> myseqs=align.seqs;
  map<string,string>::const_iterator pos;

  // Go through the entire alignment in search of
   different characters
  for (int i=0; i < al_length; i++) {
    for (pos = myseqs.begin(); pos != myseqs.end(); ++pos)
   {
      bool exclude = findChrExclude(chr_exclude, pos->
   second[i]);
        if (not exclude and alph.find(pos->second[i]) ==
   alph.end()) {
        alph[pos->second[i]] = value;
        ++value;
      }
    }
  }

  return alph;
}
```

C.2.4 realOrder

It writes the taxa names as $1, 2, 3, 4$, following the order of appearance in the FASTA file.

```
vector<int> realOrder(vector<string> taxa) {

  int nb_taxa = taxa.size();
  vector<int> order_taxa(nb_taxa);

  map<string,int> sequences;
  map<string,int>::iterator it;

  for(int i=0; i<nb_taxa; ++i) sequences[taxa[i]] = 0;
```

```
  int alph_order = 0;
  for (it = sequences.begin(); it != sequences.end(); ++it
   ) {
    sequences[it->first] = alph_order;
    ++alph_order;
  }

  for(int i=0; i<nb_taxa; ++i) {
    it = sequences.find(taxa[i]);
    order_taxa[i] = it->second;
  }

  return order_taxa;
}
```

C.2.5 realSplit

It writes the taxa names of the split as $1, 2, 3, 4$, following the order determined in function realOrder.

```
vector<int> realSplit(vector<int> split, vector<int>
   order_taxa) {

  int nb_split = split.size();
  vector<int> order_split(nb_split);

  for(int i=0; i<nb_split; ++i) order_split[i] =
   order_taxa[split[i]];

  return order_split;
}
```

C.2.6 flatteningMat

It constructs the flattening matrix of a given split.

```
vector <int> power_K;

vector<int> powersK(int numb_sp, int nb_alph) {
  int aux=1;
  for (int i=0;i<=numb_sp;i++) {
    power_K.push_back(aux);
    aux=nb_alph*aux;
  }
```

```
  return(power_K);
}

// Calculate the coordinates of the flattening matrix for
   a given column
unsigned int getCoord(string it, vector <int> set_species,
    map <char,int> alph, vector<int> power_K) {

  int nb_set = set_species.size();

  long int coord = 0;

  vector <long int> coords;
  for(int i=0; i<nb_set ; i++) {
    char nucleotide = it[set_species[i]];
    coord += alph[nucleotide]*power_K[nb_set-1-i];
  }
  return coord;
}


mat flatteningMat(map<string,int> tensor, map<char,int>
   alph, vector<int> split, vector <int> othersp, unsigned
    int seq_len) {

  unsigned int coord_row,coord_col;

  int n_rows = pow(alph.size(),split.size());
  int n_cols = pow(alph.size(),othersp.size());
  vector <int> power_K = powersK(max(split.size(),othersp.
   size()),alph.size());

  mat flatmatrix = zeros<mat>(n_rows,n_cols);

  for (map<string,int>::iterator it=tensor.begin(); it!=
   tensor.end(); ++it) {
    coord_row = getCoord(it->first,split,alph,power_K);
    coord_col = getCoord(it->first,othersp,alph,power_K);
    flatmatrix(coord_row,coord_col) = it->second;
  }
  return flatmatrix;
}
```

C.2.7 distFrobenius

It computes the Frobenius-distance between the given flattening matrix and the space of matrices with rank $\leq$ nb_partititons*nb_alph.

```
double distFrobenius(mat flat, int nb_partitions, int
    nb_alph) {

  double tail = 0;
  vec singval = svd(flat); // SVD function from Armadillo
    library

  // Frobenius distance
  for(int k=nb_alph*nb_partitions;k<singval.size();k++) {
    tail+ = singval[k]*singval[k];
  }
  return sqrt(tail);
}
```

C.3 Auxiliary software

A list of other developed software for the performance of the thesis is presented:

Software name	Description
SVD_quartets.cpp	A version of SVD software: it computes the weights of all the quartets induced by an alignment
WO_choose.cpp	A version of WO software: it chooses a random initial quartet q such that $w(q) > 0.6$
concatenate_fasta.cpp	It concatenates two alignments that are in FASTA format
permutations.cpp	It randomly permutes the columns of an alignment. It uses the so-called random library from python
subalignments.cpp	It divides an alignment in sub-alignments following the taxa groups indicated in the input
alphabet_reducts.cpp	Given an alignment, it converts the characters of the alphabet to those specified by the user. It was used to convert protein alignments to 5-state alignments (following the chemical properties) and, then, use them to test the program
remove_branch_lengths.cpp	It removes the branch lengths of a Newick tree format
bubble_chart.R	It takes the results of the Robinson-Foulds distance to plot a bubble chart (R code)

Table C.1: The main software packages developed in order to perform the thesis.